Инструкция по программированию на языке ForthLogic™

2014

Содержание

Программирование на языке ForthLogic™

Вступление

Язык программирования Форт (англ. forth вперед и одновременное сокращение от fourth четвертый), который лежит в основе языка ForthLogic™, появился в начале 1970-х гг. в США. Его изобретатель Чарльз Мур сначала применил его для разработки программного обеспечения микро-ЭВМ, которая управляла работой радиотелескопа. Преимущества работы с языком Форт были настолько большими, что вскоре его начали использовать и на других специализированных ЭВМ.

Эффективность применения языка Форт подтверждается тем, что он используется наиболее известными компаниями: корпорация Boeing использует встроенный интерпретатор языка Форт в системе авионики (бортового оборудования) самолета Boeing 777; корпорация Tektronix использует язык Форт для серии анализаторов сетевых протоколов K1297 и K1205; корпорация Lockheed Martin использует язык Форт в бортовом оборудовании наземной телеметрической системы SMART для баллистической ракеты Trident 2 D5; корпорация FedEx использует ручной считыватель штрих кодов SuperTracker со встроенной Форт-системой в своем программно-аппаратном комплексе электронного контроля посылки COSMOS II; корпорация General Electric использует язык Форт для серии SONET-коммутаторов JUNGLEMUX; корпорация Europay использует язык Форт при создании программного обеспечения для универсальных кассовых терминалов / считывателей смарт-карт с архитектурой Open Terminal Architecture (OTA); корпорация Sun Microsystems с 1989 г. использует загрузчик OpenBoot (программа типа BIOS) со встроенным интерпретатором языка Форт в своих компьютерах SparcStation и серверах SPARCServer, а корпорация Apple Inc. использует аналогичный загрузчик Open Firmware в своих компьютерах Power Macintosh. Кроме того, язык Форт является стандартным языком управления оборудованием телескопов как на земле так и в космосе.

Основные положения

Синтаксис языка ForthLogic™ максимально прост. Запись каждой инструкции (команды) состоит из одного слова, в качестве которого может выступать последовательность любых символов, которая не содержит пробела. В языке ForthLogic™ такая инструкция (команда) так и называется - *слово*. Простота синтаксиса является следствием того, что в качестве вычислительной модели используется стековая машина. В большинстве случаев слова-команды этой машины снимают необходимые операнды со стека и оставляют свои результаты (если они есть) также на стеке. Таким образом, программа, написанная на языке ForthLogic™, выглядит как последовательность слов, каждое из которых имеет в виду выполнение тех или других действий. Слова разделяются любым числом пробелов; ограничение налагается только на длину слова - оно должно содержать не больше 14 символов. Стандартно определен сравнительно небольшой набор "встроенных" слов. Среди них есть слова, которые позволяют определять новые через уже существующие и тем самым расширять начальный набор слов-команды в нужном для данного задания направлении.

Вычислительная модель, которая лежит в основе языка ForthLogic™, состоит из стека данных, стека чисел с плавающей запятой, глобальных переменных, глобальных переменных в формате чисел с плавающей запятой, глобальных битовых переменных, глобальных строчных переменных, словаря, и выходного буфера, для выведения результатов на терминал или обмена строчными данными между словами. Язык ForthLogic™ позволяет описывать параллельно выполняемые процессы и функционирует в многозадачной среде.

Стек данных (далее по тексту - или просто *стек* или *стек данных*) расположен в оперативной памяти и используется для передачи числовых параметров и результатов между словами, выполняемыми в

пределах одной задачи. Его элементами являются четырехбайтные значения, которые рассматриваются как целые числа со знаком в диапазоне от -2147483648 до +2147483647. В процессе выполнения слов значения помещаются на стек и снимаются с него. Переполнения и исчерпание стека проверяется и сообщается как ошибка; его максимальный объем равняется 16 элементам.

Стек чисел с плавающей запятой (далее по тексту - *математический стек*) также расположен в оперативной памяти и используется для математических вычислений над числами с плавающей запятой. Его элементами является четырехбайтные представления чисел с плавающей запятой одинарной точности согласно стандарту IEEE-754, которые могут принимать значение в диапазоне $\pm(1,4\text{x}10^{-45}...3,4\text{x}10^{38})$. В процессе выполнения математической операции над числами с плавающей запятой, значения помещаются на математический стек и снимаются с него. Переполнения и исчерпания математического стека проверяется и сообщается как ошибка; его максимальный объем равняется 16 элементам.

Глобальная переменная является обычной статической переменной, которая расположена в оперативной памяти. Ее элементами являются четырехбайтные значения, которые рассматриваются как целые числа со знаком. Глобальная переменная обычно используется для передачи числовых параметров и результатов целочисленных вычислений между словами выполняемыми в разных задачах или для долговременного сохранения целочисленных параметров и результатов в рамках одной задачи. Количество глобальных переменных зависит от аппаратной платформы.

Глобальная переменная в формате чисел с плавающей запятой (далее по тексту - *математическая переменная*) является статической переменной, которая расположена в оперативной памяти. Ее элементами являются четырехбайтные представления чисел с плавающей запятой одинарной точности согласно стандарту IEEE-754. Математическая переменная обычно используется для сохранения промежуточных математических значений и результатов вычислений. Количество глобальных математических переменных зависит от аппаратной платформы.

Глобальная битовая переменная является статической переменной, которая расположена в оперативной памяти. Ее элементами являются однобитовые числа которые принимают значение 0 или 1. Битовая переменная обычно используется в качестве разнообразных флажков и в задачах логического управления - она естественным образом согласовывается с унитарными сигналами контролера. Количество глобальных битовых переменных зависит от аппаратной платформы.

Глобальная строчная переменная является статическим массивом длиной 15 элементов, который расположен в оперативной памяти. Элементами массива являются однобайтовые значения, которые представляют коды символов. Строчная переменная предназначена для оперативного хранения строк текста длиной до 15 символов, в которых могут присутствовать пробелы. Количество глобальных строчных переменных зависит от аппаратной платформы.

Начальную часть энергонезависимой памяти занимает *словарь* - хранилище слов, констант и строчных данных. По мере расширения начального набора слов, словарь растет в сторону увеличения адресов. Существуют специальные слова, которые позволяют работать со словарем (например, выяснять свободное место в словаре, удалять слова и тому подобное). Объем энергонезависимой памяти и размер начального словаря зависит от аппаратной платформы.

Выходной буфер является обычным статическим буфером, который расположен в оперативной памяти. Его элементами являются однобайтовые значения, которые представляют коды символов. В буфере располагаются строчные данные для обмена с терминалом или между отдельными словами. Размер буфера зависит от аппаратной платформы.

Работа в диалоговом режиме

Программирование на языке ForthLogic™ является диалоговым процессом. Работая за терминалом, пользователь вводит слова-команды, а *форт-система*, то есть программно-аппаратная реализация языка ForthLogic™, немедленно выполняет действия, которые обозначаются этими словами, - это *терминальный режим* работы форт-системы. О своей готовности к обработке следующей строки текста, форт-система сообщает пользователю приглашением ">", которое печатается на терминале. Получив такое приглашение, пользователь набирает на терминале следующую порцию текста, заканчивая ее клавишей "Enter". Получив сигнал о завершении введения, форт-система начинает обработку введенного текста (он размещается во входном буфере для введения с терминала), выделяя в нем слова-команды и выполняя их. Успешно обработав весь введенный текст, форт-система опять приглашает пользователя к введению и описанный цикл диалога повторяется. После успешного завершения обработки очередного введенного текста, форт-система выводит на терминал подтверждающее сообщение (ОК) (от английского o'kay - "все в порядке"). Если во время обработки введенного текста происходит любая ошибка (например, встретилось неизвестное форт-системе слово), то на терминал выводится объясняющее сообщение, обработка введенного текста, прекращается и форт-система приглашает пользователя к введению нового текста. Все известные форт-системе ошибки и их коды описаны в дополнении.

Максимальная длина строки текста, которая принимается для обработки, составляет 80 символов. При работе с терминалом, встроенный редактор форт-системы автоматически ограничивает длину строки до 77 символов и извещает об этом звуковым сигналом. При наборе очередной порции текста, кроме клавиш букв и цифр, допускается применять лишь клавишу "Backspace", "Space" (пробел) и "Enter".

Под словом терминал мы понимаем любую из доступных программ, которые осуществляют эмуляцию терминала. В среде разных операционных систем таких программ очень много, причем как платных так и бесплатных - процедура настройки программы-терминала в среде Microsoft® Windows®XP описана в дополнении.

Непосредственно вводить с терминала большие тексты неудобно, потому их сохраняют в текстовые файлы и с помощью карты памяти SD/MMC переносят в форт-систему (процедура создания файлов и переноса их на карту памяти SD/MMC описанная в дополнении). Также текстовый файл можно перенести в форт-систему с помощью протокола Xmodem непосредственно из терминала - данная процедура также описана в дополнении.

Контролеры без встроенного дисплея имеют специальный режим отображения статусной информации о форт-системе в окне терминала. Для входа и выхода из этого режима существуют соответственно команды SHOW STATUS и HIDE STATUS, которые следуют вводить с новой строки терминала.

Стек данных и вычисления

Как уже упоминалось, в основе вычислительной модели языка ForthLogic™ лежит стековая машина. На данный момент, форт-система содержит два программно доступных стека: стек данных и математический стек. Механизмы работы этих стеков подобны - разница заключается в представлении данных и в наборе слов для работы с данными на этих стеках.

Стек представляет собой программный объект, который функционирует по принципу "последний зашел - первый вышел". Стек является единственным местом проведения математических и логических вычислений и является чрезвычайно простым и надежным механизмом передачи числовых параметров и данных между отдельными словами. В качестве аналогии со стеком можно привести колоду карт: положить новое значение на стек можно сравнить с тем, как мы кладем новую карту на верх колоды; снять число со стека аналогично действию, когда мы снимаем карту с вершины колоды. Слова на языке ForthLogic™ обычно используют в качестве операндов верхние элементы

стека, убирая их со стека и возвращая результаты (если они есть) на место операндов. Как правило, слова используют одно - два верхних значения на стеке. Для их описания будем применять следующую диаграмму стековой нотации и цветное выделение (это касается обоих стеков):

```
 имя          вершина стека до  --->  вершина стека после
 слова        выполнения слова        выполнения слова
```

При этом считаем, что самое верхнее значение на стеке (то, которое было добавлено последним) находится справа. Дополнительно, перед значениями, которые находятся на математическом стеке, будем указывать тип стека с помощью буквы *F* и двоеточия.

Для работы с вершиной стека данных существуют следующие слова:

```
    DUP       A         --->      A,A
    DROP      A         --->      -
    OVER      A,B       --->      A,B,A
    ROT       A,B,C     --->      B,C,A
    SWAP      A,B       --->      B,A
```

Слово DUP (от DUPLICATE дублировать) дублирует вершину стека данных, добавляя на стек еще одно значение, ровное тому, которое было верхним. Слово DROP (сбросить) убирает верхнее значение. Слово OVER (через) дублирует значение, которое расположено на стеке данных непосредственно под верхним. Слово ROT (от ROTATE вращать) циклически переставляет по часовой стрелке три верхних значений на стеке данных. Наконец, слово SWAP (обменять) меняет местами два верхних значения.

Можно также работать с любым элементом стека данных с помощью следующих слов:

```
    PICK      An,An-1...,Ao,n --->    An,An-1...,Ao,An
    ROLL      An,An-1...,Ao,n --->    An-1...,Ao,An
```

Слово PICK (взять) дублирует n-й элемент стека данных (считая от нуля). Так образом, 0 PICK тождественно DUP, а 1 PICK тождественно OVER. Слово ROLL (повернуть) циклически переставляет n верхних элементов стека (тоже считая от нуля) по часовой стрелке, так что 2 ROLL тождественно ROT, 1 ROLL тождественно SWAP, а 0 ROLL является пустой операцией.

Чтобы "увидеть" верхнее значение на стеке данных, используется слово **.** (точка), которое снимает значение с вершины стека и печатает его в выходном буфере и на терминале как целое число в свободном формате (то есть без ведущих нулей и со знаком минус, если число негативное). Если пользователь хочет, чтобы напечатанное значение осталось на стеке, он должен выполнить следующий текст:

```
 > DUP .
```

Слово DUP создаст копию верхнего значения, а точка его распечатает и уберет со стека.

Для работы с вершиной математического стека существуют следующие слова:

```
    FDUP      F:A       --->      F:A,A
```

```
FDROP     F:A        --->      F: -
FOVER     F:A,B      --->      F:A,B,A
FROT      F:A,B,C    --->      F:B,C,A
FSWAP     F:A,B      --->      F:B,A
```

Слово FDUP (от FLOAT DUPLICATE дублировать) дублирует вершину математического стека, добавляя на стек еще одно значение, ровное тому, которое было верхним. Слово FDROP (сбросить) убирает верхнее значение. Слово FOVER (через) дублирует значение, которое расположено на математическом стеке непосредственно под верхним. Слово FROT (от FLOAT ROTATE вращать) циклически переставляет по часовой стрелке три верхних значений на математическом стеке. Наконец, слово FSWAP (обменять) меняет местами два верхних значения.

Также можно работать с любым элементом математического стека с помощью слов:

```
FPICK     F:An,An-1...,Ao,n --->   F:An,An-1...,Ao,An
FROLL     F:An,An-1...,Ao,n --->   F:An-1...,Ao,An
```

Слово FPICK (взять) дублирует n-й элемент математического стека (считая от нуля), так что 0 FPICK тождественно FDUP, а 1 FPICK тождественно FOVER. Слово FROLL (повернуть) циклически переставляет n верхних элементов математического стека (тоже считая от нуля) по часовой стрелке, так что 2 FROLL тождественно FROT, 1 FROLL тождественно FSWAP, а 0 FROLL является пустой операцией.

Чтобы "увидеть" верхнее значение на математическом стеке, используются слова F**.** (FLOAT - точка) или FE**.** (FLOAT ENGINEER - точка), которые снимают значение с вершины математического стека и печатают его в выходном буфере и на терминале. В первом случае число печатается с фиксированной запятой в свободном формате с шестью знаками после запятой (то есть без ведущих нулей и со знаком минус, если число негативное). Во втором случае число печатается в инженерном / научном представлении с мантиссой, основой 10 и экспонентой (например -1,234E-02; 1,98E+12). Точность представления чисел при распечатке (то есть количество цифр после десятичной запятой) можно установить с помощью системной переменной FPREC, которая по умолчанию равна 6. Для записи нового значения в системную переменную FPREC существует слово FPREC!, которое работает следующим образом: с вершины стека данных снимается число (новое значение переменной FPREC в диапазоне от 0 до 6) и присваивается этой переменной. Точность равная 0 означает отсутствие десятичной запятой вообще, что тождественно превращению к целому. Значение системной переменной FPREC не сохраняется при выключении питания, потому, при использовании другой чем по умолчанию точности представления чисел при распечатке, данную переменную следует инициализировать.

Как уже упоминалось, стек представляет собой структуру данных, которая функционирует по принципу "последний зашел - первый вышел". Для построения некоторых алгоритмов обработки данных такая структура может пригодиться. Необходимо знать количество элементов данных на стеке, чтобы иметь возможность выполнить цикл обработки этих данных определенное количество раз. Для определения количества элементов на стеке данных и математическом стеке, существуют слова DEPTH (глубина) и FDEPTH (FLOAT глубина), которые кладут на вершину стека данных количество элементов на стеке данных и на математическом стеке соответственно:

```
DEPTH     An,An-1...,A1  --->      An,An-1...,A1,n

FDEPTH    -              --->      n
```

```
F:An,An-1...,A1 --->     F:An,An-1...,A1
```

Перечисленные выше слова работают со значениями, которые уже находятся на стеке. А как занести значение на стек? Язык ForthLogic™ имеет следующее *правило по умолчанию:* если введенное слово форт-системе не известно, то прежде чем сообщать пользователю об ошибке, форт-система пытается понять это слово как запись числа. Если слово-число состоит из одних только цифр с возможным начальным знаком минус, то ошибки нет: слово считается известным и его действие заключается в том, что данное число кладется на вершину стека данных. Если слово-число состоит из одних цифр разделенных десятичной запятой (в действительности это символ точка .) с возможным начальным знаком минус и возможным научным представлением экспоненты числа (с помощью символов е или Е и возможным знаком экспоненты минус), то слово также считается известным и его действие заключается в том, что данное число кладется на вершину математического стека.

Для непосредственного переноса чисел с одного стека на другой существуют слова D>F и F>D. Слово D>F снимает верхнее значение с вершины стека данных и переносит на вершину математического стека с соответствующим превращением представления числа. Слово F>D снимает верхнее значение с вершины математического стека и переносит на вершину стека данных с округлением к ближайшему целому числу и соответствующим превращением представления числа. В стековой нотации для данных слов (и других подобных им) необходимо отображать состояние обоих стеков:

```
D>F      A        ---> -
         F: -     ---> F: преобразованное значение A

F>D      F:A      ---> F: -
         -        ---> округленное значение A
```

Теперь у нас достаточно средств, чтобы привести примеры диалога. Рассмотрим следующий протокол работы (далее по тексту будем выделять протокол работы с терминалом серым цветом):

```
> 5 6 7
(OK)
> SWAP . . .
6 7 5 (OK)
> 123.456 -12.987E-2 FE. F.
1.234560E+02 -0.129870 (OK)
>
```

В ответ на приглашение к введению (символ ">" напечатанный системой), пользователь вводит три числа: 5, 6 и 7. Обрабатывая введенный текст, форт-система кладет эти числа в указанном порядке на стек данных и по окончании обработки выводит подтверждающее сообщение ОК и снова приглашает пользователя к введению. Дальше пользователь вводит текст из четырех слов: SWAP и три точки. Выполняя эти слова-команды, форт-система меняет местами два верхних элементов стека данных (5, 6, 7 -> 5, 7, 6) и потом по очереди три раз снимает верхнее значение со стека данных и печатает его. В результате на терминале появляется текст 6 7 5 и сообщение ОК, указывающее о завершении обработки, после чего система снова выдает пользовательское приглашение на введение. В ответ на приглашение к введению пользователь вводит два числа с фиксированной и плавающей запятой и слова для отображения этих значений с вершины математического стека в разном формате.

Обрабатывая введенный текст, форт-система кладет эти числа в указанном порядке на математический стек, затем снимает их со стека, печатая при этом в указанном формате.

В предыдущем примере, при выведение значений со стека на терминал (а также в выходной буфер), автоматически печатается один пробел. Такое поведение форт-системы устанавливается по умолчанию, однако можно произвольным образом активировать или дезактивировать автоматическую печать одного пробела после любого вывода на терминал (а также в выходной буфер). Для этого существуют слова AUTOSPACE - активация автоматической печати одного пробела, и NOAUTOSPACE - дезактивация автоматической печати одного пробела.

Чтобы увеличить количество пробелов (для форматирования вывода) можно применить слово SPACE (пробел). Данное слово печатает в выходном буфере и на терминале один пробел:

```
> NOAUTOSPACE 5 6 7
(OK)
> SWAP . SPACE . SPACE . SPACE
6 7 5 (OK)
> AUTOSPACE 5 6 7
(OK)
> . . .
7 6 5 (OK)
>
```

Чтобы перенести текст на новую строку (для форматирования вывода) можно применить слово NEWLINE (новая строки). Данное слово переносит текст в выходном буфере и на терминале на новую строку, однако практическое применение этого слова имеет смысл при выведенные текста на встроенный дисплей и для форматирования текстов SMS (см. следующие разделы).

Слова-команды, которые выполняют целочисленные арифметические операции над числами со стека данных, являются общепринятыми математическими обозначениями:

```
+         A,B       ---> сумма A+B
-         A,B       ---> разница A-B
*         A,B       ---> произведение A*B
/         A,B       ---> частное от деления A/B
MOD       A,B       ---> остаток от деления A/B
ABS       A         ---> абсолютная величина A
NEGATE    A         ---> значение с обратным знаком -A
```

Слова-команды, которые выполняют математические операции над числами в формате с плавающей запятой на математическом стеке, отличаются от общепринятых добавлением буквы F:

```
F+        F:A,B     ---> F: сумма A+B
F-        F:A,B     ---> F: разница A-B
F*        F:A,B     ---> F: произведение A*B
F/        F:A,B     ---> F: деление A/B
FABS      F:A       ---> F: абсолютная величина A
FNEGATE   F:A       ---> F: значение с обратным знаком -A
```

Слова-команды, которые выполняют вычисление математических функций над числами в формате с плавающей запятой на математическом стеке:

```
     FSIN       F:A        ---> F: синус угла A в радианах sin(A)
     FCOS       F:A        ---> F: косинус угла A в радианах cos(A)
     FTAN       F:A        ---> F: тангенс угла A в радианах tg(A)
     FSINH      F:A        ---> F: синус гиперболический A sh(A)
     FCOSH      F:A        ---> F: косинус гиперболический A ch(A)
     FTANH      F:A        ---> F: тангенс гиперболический A th(A)
     FASIN      F:A        ---> F: арксинус A arcsin(A),-1.0<=A<=1.0
     FACOS      F:A        ---> F: арккосинус A arccos(A),-1.0<=A<=1.0
     FATAN      F:A        ---> F: арктангенс A arctg(A)
     FLOG       F:A        ---> F: логарифм десятичный A log10(A)
     FLN        F:A        ---> F: логарифм натуральный A ln(A)
     FEXP       F:A        ---> F: экспонента A eA
     F**        F:A,B      ---> F: возведение A в степень B A^B
     FSQRT      F:A        ---> F: корень квадратный из A √A
```

Использование стека для хранения промежуточных значений естественным образом приводит к так называемой "обратной польской форме" - одному из способов бесскобочной записи арифметических выражений, что имеет в виду постановку знака операции после операндов. Например, выражение (A/B+C)*(D*E-F*(G-H)) записывается таким образом: A B / C + D E * F G H - * - *. Таким образом, форт-систему можно использовать в качестве калькулятора целочисленных значений. Чтобы вычислить, например, значение (25+18+32)*5, достаточно ввести такой текст:

```
> 25 18 + 32 + 5 * .
375 (OK)
>
```

В ответ система напечатает (выполняя "точку") желаемый ответ. Также, форт-систему можно использовать и в качестве калькулятора для чисел с плавающей запятой. Чтобы вычислить значение 1.2e-2*sin(25.23+0.18*3.1415), можно ввести текст:

```
> 1.2e-2 25.23 0.18 3.1415 F* F+ FSIN F* F.
0.007383 (OK)
>
```

Введение новых слов

Базовое свойство языка ForthLogic™ - это возможность вводить новые слова, расширяя тем самым набор команд в нужном для пользователя направлении. Для введения новых слов чаще всего используется *определение через двоеточие* - определение нового слова через уже известные форт-системе слова. Такое определение начинается словом : (двоеточие) и заканчивается словом ; (точка с запятой). Сразу после двоеточия идет слово, которое определяется, а за ним последовательность

слов, через которые оно определяется. Слово которое определяется должно содержать не больше 14 любых символов и не содержать пробелов. Например:

```
> : S2 DUP * SWAP DUP * + ;
(OK)
>
```

определяет слово S2, которое вычисляет сумму квадратов двух чисел, которые снимаются с вершины стека, стековая нотация при этом имеет вид:

```
   S2          A,B        --->       A**2+B**2
```

После введения данного описания, слово S2 можно выполнять и включать в описания других слов. При создании таких определений рекомендуется тщательным образом отслеживать все изменения стека.

Перепишем приведенное выше определение слова S2 с комментариями, показывая в скобках состояние вершины стека после выполнения каждой строки. Слово ((левая скобка) предназначено для комментариев - форт-система игнорирует все слова после левой скобки вплоть до конца строки или до первой закрывающей правой скобки, перед которой должен быть по крайней мере один пробел:

```
: S2          ( A,B ---> A**2+B**2 сумма квадратов )
  DUP         ( A,B,B )
  * SWAP      ( B**2,A )
  DUP *       ( B**2,A**2 )
  +           ( A**2+B**2 )
;
(OK)
>
```

Рассмотрим подробнее работу форт-системы во время определения новых слов. Мы уже знаем, что получив от пользователя очередную порцию входного текста, форт-система выделяет в ней отдельные слова и ищет их в своем словаре. Эту работу выполняет встроенный текстовый интерпретатор форт-системы. Если слово в словаре не найдено, то текстовый интерпретатор пытается понять его как число, используя описанное выше правило по умолчанию. Если слово найдено или оказались записью числа, то последующие действия интерпретатора зависят от его текущего состояния. В каждый момент времени текстовый интерпретатор находится в одном из двух состояний: в состоянии *выполнения* или в состоянии *компиляции*. В состоянии выполнения найденное слово выполняется (то есть выполняется действие, которое составляет его семантику), а число кладется на стек. Если же интерпретатор находится в состоянии компиляции, то найденное слово не выполняется, а компилируется, то есть включается в создаваемую последовательность действий для слова, которое определяется в данный момент. Найденное и скомпилированное таким образом слово будет выполнено вместе с другими такими словами во время выполнения определенного через них слова. Если нужно скомпилировать число, то текстовый интерпретатор компилирует особый код, который во время выполнения положит значение данного числа на стек (это действие абсолютно прозрачно для пользователя). Скомпилированные слова выполняются чрезвычайно быстро: разница

между временем интерпретации определенной последовательности действий и временем выполнения той же скомпилированной последовательности действий достигает трех порядков - это огромное преимущество языка ForthLogic™ над другими интерпретирующими языками.

Проследим за работой текстового интерпретатора по обработке уже рассмотренного определения слова S2. Допустим, что перед началом обработки введенной строки интерпретатор находится в состоянии выполнения. Первым словом является : (двоеточие), которое выполняется. Его семантика заключается в том, что из входной строки выбирается очередное слово и запоминается в словаре как слово, которое в данный момент определяется, а интерпретатор переключается в состояние компиляции. Следующие слова, которые интерпретатор будет выбирать из входной строки (DUP *, SWAP, и тому подобное), не учитывая комментариев, будут компилироваться а не выполняться, поскольку интерпретатор находится в состоянии компиляции. В результате, со словом S2 связывается последовательность действий, которая отвечает этим словам. Процесс выделения и компиляции слов будет продолжаться до тех пор, пока не встретится ; (точка с запятой). Это слово особенное, оно имеет так называемый признак *немедленного выполнения.* Слова с таким признаком выполняются независимо от текущего состояния текстового интерпретатора, поэтому точка с запятой будет вторым выполненным словом после двоеточия. Семантика точки с запятой заключается в том, что построение определения, начатого двоеточием, завершается и интерпретатор снова переключается в состояние выполнения. Поэтому, после введения определения слова S2, мы сразу можем проверить как оно работает на конкретных значениях:

```
> 5 4 S2 .
41 (OK)
>
```

Слова с признаком немедленного выполнения выполняются по-разному в зависимости от состояния текстового интерпретатора. Некоторые из них даже используются только в одном из этих состояний. Например, слово ; допустимо применять только в состоянии компиляции. Оно завершает построение нового определения и переключает текстовый интерпретатор в состояние выполнения.

Введенные слова можно исключить из словаря с помощью слова FORGET (забыть), которое выбирает из входной строки следующее слово (введенное после слова FORGET) и исключает его из словаря вместе со всеми словами, определенными позже. Такой способ исключения слов выглядит естественным при описании задачи методом "снизу-вверх", который используется в языке ForthLogic™.

Разберем следующий протокол диалога:

```
> 2 2 * .
4 (OK)
> : 2 3 ;
(OK)
> 2 2 * .
9 (OK)
> FORGET 2
(OK)
> 2 2 * .
4 (OK)
>
```

Сначала пользователь вычисляет результат от умножения 2 на 2 и получает ответ 4. Введя после этого определение слова 2 как число 3, он в дальнейшем получает уже другой ответ. Исключив это определение слова 2 через FORGET, он возвращается к прежней семантике слова 2 - то есть просто цифра.

Для работы с новыми словами и словарем, существуют также слова WORDS (слова) и UNUSED (не использовано). Слово WORDS печатает на терминале список всех доступных слов в словаре в порядке их определения (если список не помещается на экране, то делается остановка, пока пользователь не нажмет клавишу "Space" (пробел)). Слово UNUSED кладет на стек свободное место в словаре в байтах. Дальше, с помощью слова . (точка), это значение можно отобразить на терминале. Также существует возможность возвращения словаря к первичному состоянию, когда в нем присутствуют лишь встроенные слова. Для этого, в терминальном режиме работы текстового интерпретатора форт-системы необходимо с новой строки указать команду BUILD DICTIONARY. Данная команда полностью перестраивает словарь на основе внутренней информации и применяется в случае серьезных сбоев в работе встроенных слов. Для простого вытирания всех слов пользователя существует слово ERASE (вытереть).

Конкретное содержимое словаря зависит от версии программной прошивки. Узнать о версии можно с помощью слова VERSION (версия) которое печатает в Выходной буфер и на терминале строку версии данной программной прошивки.

При определении новых слов через двоеточие, иногда случаются так, что не хватает места в пределах одной строки, чтобы закончить построение нового слова. В таких случаях, когда строки превышает установленную длину, допускается переходить на другую строку с помощью клавиши "Enter" и продолжать определение в новой строке. Этот процесс можно повторять вплоть до окончания определения - то есть до слова ; (точка с запятой), при этом форт-система не должна выполнять никаких посторонних заданий или программ.

При определении новых слов через двоеточие, можно применять как встроенные слова форт-системы так и другие предварительно определенные через двоеточие слова. Глубина вложений определений через двоеточие может достигать 64 уровней и тесно связана с программно-недоступным *стеком возвратов*. Стек возвратов используется для временного хранения адресов и некоторой служебной информации. Во время выполнения определения любого слова, на вершине стека возвратов будет находиться адрес того слова, из которого данное определение было вызвано (*адрес возвратов*). Адреса возвратов кладутся на стек возвратов и снимаются с него абсолютно прозрачно для пользователя, однако, переполнения и исчерпания стека возвратов проверяется и сообщается как ошибка.

Общая идеология использования собственных слов при написании программы на языке ForthLogic™ заключается в том, что сначала создаются сами простые "базовые" слова которые оперируют наименьшей порцией данных или выполняют самые простые действия. Их роботу обычно можно немедленно проверить и откорректировать в терминальном режиме путем имитации "рабочего" окружения (подставив на стек или в выходной буфер необходимые данные, подав на вход необходимый сигнал, и тому подобное). Убедившись в корректности этих слов, на их основе можно создавать и проверять более сложные слова, на их основе еще более сложные и так далее. Этот процесс в конце сводится к созданию собственного словаря задачи и одного *главного слова*, которое определяет и запускает на выполнение всю программу. Понятно, что таких программ в словаре может быть несколько и каждая из них обязательно имеет собственное главное слово. Выполнение того или иного главного слова может полностью изменить поведение системы. Данный факт является уникальным в среде программируемых контролеров. Также важен упомянутый выше способ написания программы на языке ForthLogic™ снизу-вверх который, к тому же, является единственно возможным. Такая программа содержит наименьшее возможное количество ошибок.

Константы, строки и переменные

Пользователю часто бывает удобно работать не с "анонимными" значениями, а с поименованными. По аналогию со средствами других языков эти средства языка ForthLogic™ называются константами и переменными. Также существуют строки - объекты, которые используются для выведения текста в выходной буфер и на терминал.

Константа, как следует из ее названия, является поименованной неизменной величиной. Физически константа располагается в словаре среди других слов, а поскольку словарь находится в энергонезависимой памяти то и константа также имеет свойство сохранять свое значение при выключении питания. Однако существует возможность менять значение константы, что позволяет реализовать долговременное энергонезависимое хранение любых числовых параметров конкретной задачи. Нужно лишь помнить, что для контролеров первых серий а также серии M количество изменений одной константы ограничено на уровне, приблизительно, 10^5 раз и сам процесс изменения является достаточно долговременным (4-10 мсек). Для остальных контролеров количество изменений одной константы ограничено на уровне, приблизительно, 10^{10} раз а процесс записи соизмерим по времени с процессом чтения и может не приниматься во внимание. Константы могут быть *обычные* - это те которые были определены пользователем и могут быть переопределены, и *системные* - это константы из базового набора слов, которые упрощают выполнение определенных заданий и не могут быть переопределены.

Рассмотрим слова для работы с константой. Слово CONSTANT (константа) работает следующим образом. Со стека данных снимается верхнее значение, а из входного текста выбирается очередное слово (введенное после слова CONSTANT) и запоминается в словаре как новая команда - название константы целого типа. Ее действие заключается в следующем: положить на стек данных значение, снятое со стека в момент ее определения. Слово TO (в) работает следующим образом. Со стека данных снимается верхнее значение, а из входного текста выбирается очередное слово (введенное после слова TO) - название константы целого типа. Дальше осуществляется модификация данной константы в соответствии с введенным значением.

Слово FCONSTANT (FLOAT-константа) работает аналогичным образом - только лишь касается математического стека. С математического стека снимается верхнее значение, а из входного текста выбирается очередное слово (введенное после слова FCONSTANT) и запоминается в словаре как новая команда - название константы плавающего типа. Ее действие заключается в следующем: положить на математический стек значение, снятое со стека в момент ее определения. Слово TOF (в) работает следующим образом. С математического стека снимается верхнее значение, а из входного текста выбирается очередное слово (введенное после слова TOF) - название константы плавающего типа. Дальше осуществляется модификация данной константы в соответствии с введенным значением.

Названия констант должны содержать не больше 14 любых символов и не содержать пробелов. Рассмотрим пример:

```
> 4 CONSTANT ХОРОШО 3.14156 FCONSTANT PI
(OK)
> ХОРОШО . PI F.
4 3.141560 (OK)
> 5 TO ХОРОШО ХОРОШО .
5 (OK)
>
```

В дальнейшем, при выполнение слова ХОРОШО, на стек данных будет положено число 5, а при выполнении слова PI - на математический стек будет положено число 3,14156.

Кроме одиночной константы существует возможность создавать и работать с массивом констант целого типа. Провозглашение массива констант осуществляется с помощью слова ARRAY (массив) следующим образом. Со стека данных снимается верхнее значение - размер провозглашаемого массива, а из входного текста выбирается очередное слово - имя массива (введенное после слова ARRAY) и запоминается в словаре как новая команда. Ее действие заключается в следующем: положить на стек данных значения которое отвечает определенному элементу массива. Во время провозглашения все члены массива принимают нулевые значения. Существует ограничение на максимальный размер провозглашаемых массивов - 256. Нумерация элементов массива начинается от числа 1. Имя массива должно содержать не больше 14 любых символов и не содержать пробелов.

Для доступа к элементам массива используется номер элемента, который снимается со стека данных, и само имя массива. При этом на стек возвращается значение элемента массива. Для изменения элемента массива существует слово SET (установить), которое работает следующим образом: со стека данных снимается два верхних значения - номер элемента и новое значение элемента, а из входного текста выбирается очередное слово - имя массива (введенное после слова SET). Дальше осуществляется модификация данного элемента массива.

Также для массивов реализована операция поиска первого от начала элемента массива равного определенному значению - слово FINDINDEX. Оно работает следующим образом: со стека данных снимается верхнее значение - величина которая ищется, а из входного текста выбирается очередное слово - имя массива (введенное после слова FINDINDEX), дальше осуществляется поиск и на стек возвращается первый от начала номер элемента массива равного заданной величине. Если поиск не удачен то на стек возвращается нулевое значение.

Переменная это поименованная ячейка оперативной памяти. Физическая природа этой памяти позволяет осуществлять модификацию переменной бесконечное количество раз, процесс модификации чрезвычайно быстр (~100 нсек), однако оперативная память не есть энергонезависимой.

Пользователю доступны *D_MAX* глобальных переменных для хранения целых чисел имена которых - это номера в диапазоне от 1 до D_MAX. Количество глобальных переменных D_MAX зависит от аппаратной платформы на которой функционирует форт-система и приведено в применении. Для работы с глобальными переменными существуют слова VAR? и VAR!. Слово VAR? работает следующим образом. Со стека данных снимается верхнее значение - номер переменной, а на стек данных кладется копия содержимого соответствующей переменной. Слово VAR! работает иным образом. Со стека данных снимается два верхних значения - номер переменной и новое значение, которое запоминается в соответствующей переменной. Приведем пример работы с этими словами:

```
> 4 1 VAR! -237889 16 VAR!
(OK)
> 1 VAR? .
4 (OK)
> 16 VAR? .
-237889 (OK)
>
```

Пользователю также доступные F_MAX глобальных математических переменных для хранения чисел в формате с плавающей запятой имена которых - это номера в диапазоне от 1 до F_MAX. Количество глобальных математических переменных F_MAX зависит от аппаратной платформы на которой

функционирует форт-система и приведено в применении. Для работы с глобальными математическими переменными существуют слова FVAR? и FVAR!. Слово FVAR? работает следующим образом. Со стека данных снимается верхнее значение - номер математической переменной, а на математический стек кладется копия содержимого соответствующей переменной. Слово FVAR! работает иным образом. Со стека данных снимается верхнее значение - номер переменной, с математического стека снимается верхнее значение и запоминается в соответствующей математической переменной. Приведем пример работы с данными словами:

```
> 4.123 1 FVAR! -2.37889 16 FVAR!
(OK)
> 1 FVAR? F.
4.123000 (OK)
> 16 FVAR? FE.
-0.237889E+01 (OK)
>
```

Целочисленная и математическая переменная имеют достаточно широкий диапазон представления чисел и, соответственно, занимают в памяти достаточно много места. Однако, для задач управления часто необходимо оперировать отдельными битами и флажками. Конечно, для этого можно использовать целые числа и целочисленные переменные, но это неэффективно с точки зрения затрат памяти и времени выполнения. Для работы с отдельными битами и флажками пользователю доступны B_MAX глобальных битовых переменных имена которых - это номера в диапазоне от 1 до B_MAX. Количество глобальных битовых переменных B_MAX зависит от аппаратной платформы на которой функционирует форт-система и приведено в применении. Для работы с битовыми переменными существуют слова FLAG? и FLAG!. Слово FLAG? работает следующим образом. Со стека данных снимается верхнее значение - номер переменной, а на стек данных кладется копия содержимого соответствующей переменной, которая принимает значение ИСТИНА или НЕ ИСТИНА (с данными определениями можно ознакомиться в параграфе "Логические операции"). Слово FLAG! работает иным образом. Со стека данных снимается два верхних значения - номер битовой переменной и новое значение, которое интерпретируется как ИСТИНА (для чисел не равных нулю) или НЕ ИСТИНА (для чисел равных нулю) и запоминается в соответствующей битовой переменной в виде 1 или 0. Приведем пример работы с данными словами:

```
> 1 1 FLAG! 0 16 FLAG! 124 100 FLAG!
(OK)
> 1 FLAG? . 16 FLAG? . 100 FLAG? .
-1 0 -1 (OK)
>
```

Строки текста помогают пользователю при программировании специфических команд, таких как осуществление и прием голосовых звонков, воспроизведение звуковых файлов, выведение на дисплей и тому подобное. Эти команды используют форматированное содержимое выходного буфера. При работе в терминальном режиме форт-системы, выведение в выходной буфер одновременно дублируется выведением на терминал. Размер выходного буфера составляет *OUTBUF_MAX*. При переполнении выходного буфера происходит автоматическое отбрасывание данных которые в него не помещаются и это не вызывает никакой ошибки.

Для определения строки текста которая будет печататься в выходном буфере, используется слово ." (точка-кавычка) после которого вводится любой необходимый текст с возможными пробелами. В конце текста необходимо ввести пробел и слово " (кавычка), или можно просто закончить текст клавишей "Enter". Определение строки текста с помощью слов ." (точка-кавычка) и " (кавычка) не позволяет применять в тексте саму кавычку если перед ней должен быть пробел. Если возникает необходимость печатать в выходном буфере текст с кавычкой, то для этого можно использовать слово QUOTE (кавычка). Приведем пример:

```
> NOAUTOSPACE ." Hello " SPACE QUOTE ." WORLD " QUOTE
> Hello "WORLD"(OK)
>
```

Такое определение строки текста можно использовать как внутри определения через двоеточие, что имеет большое практическое значение, так и в диалоговом режиме. строки текста, определенная внутри определения через двоеточие, имеет признак константной строки, не подлежит последующей модификации и есть энергонезависимой. Приведем примеры использования строк:

```
> ." +380501234567 "
+380501234567 (OK)
> : Hi! ." Hello world! " ;
(OK)
> Hi!
> Hello world! (OK)
> ." Вас приветствует система X
Вас приветствует система X (OK)
>
```

Для определения того что строки текста в выходном буфере не является пустой, существует слово LENGTH (длина), которое кладет на стек длину строки текста (включительно с пробелами), который находится в выходном буфере. В отличие от других слов, которые используют содержимое выходного буфера и *очищают* его при этом, слово LENGTH оставляет выходной буфер без изменений. Для непосредственной очистки выходного буфера применяется слово FLUSH (смывать).

Для оперативного хранения строки символов длиной до 15 символов в оперативной памяти существуют статические строчные переменные. Пользователю доступны *S_MAX* строчных переменных имена которых - это номера в диапазоне от 1 до S_MAX. Количество строчных переменных S_MAX зависит от аппаратной платформы на которой функционирует форт-система и приведено в применении. Для работы со строчными переменными существуют слова STRING? и STRING!. Слово STRING? работает следующим образом. Со стека данных снимается верхнее значение - номер переменной и в выходной буфер печатается содержимое данной переменной. Слово STRING! работает обратном образом. Со стека данных снимается верхнее значение - номер переменной а из выходного буфера считывается строки символов и запоминается в соответствующей переменной. Содержимое выходного буфера при этом очищается. Приведем пример работы с этими словами:

```
> ." +380501234567 "   1 STRING!
+380501234567 (OK)
```

```
> 1 STRING?
+380501234567 (OK)
>
```

Имена переменных в виде целых чисел естественным образом позволяют реализовать *индексный доступ* к ячейкам памяти разного типа. В частности, организация массивов является естественным применением такого доступа. Рассмотрим следующий пример: необходимо организовать массив из 10 целых чисел с соответствующим доступом к нему. Массив расположим начиная с номера переменной записанной в константе base:

```
> 15 CONSTANT base
(OK)
> : readarray base + VAR? ; ( n ---> A(n))
(OK)
> : writearray base + VAR! ; ( A(n),n ---> - )
(OK)
> 123 4 writearray
(OK)
> 1 readarray 4 readarray
0 123 (OK)
>
```

В этом примере мы использовали косвенную адресацию переменных, когда номер переменной внутри массива определялся путем математических вычислений. Номер переменной любого типа также может сохраняться в целочисленной переменной и это дает еще одно преимущество индексного доступа к оперативным данным.

Логические операции

В языке ForthLogic™ число 0, в двоичном эквиваленте которого все разряды нули, представляет логическое значение НЕ ИСТИНА, а любое другое 32-разрядное число воспринимается как ИСТИНА. Вместе с тем стандартные слова, которые должны возвращать в качестве результата логическое значение, из всех возможных представлений значения ИСТИНА используют только одно: число -1, в двоичном эквиваленте которого все разряды единицы. Это связано с тем, что логические операции конъюнкции, дизъюнкции и отрицания, выполняются в языке ForthLogic™ *поразрядно* над всеми разрядами операндов и в этом случае соответственно трактуются как *поразрядные логические операции.* Однако, их можно трактовать и как традиционные *логические операции*, но только над величинами ИСТИНА и НЕ ИСТИНА в упомянутом выше представлении. Для удобства, при осуществлении традиционных логических операций существует специальные системные константы TRUE (истина) и FALSE (не истина), которые кладут на стек логические значения ИСТИНА и НЕ ИСТИНА соответственно:

```
    TRUE      -     ---> -1
    FALSE     -     ---> 0
```

Слова-команды, которые выполняют логические вычисления над значениями стека данных имеют традиционный синтаксис и работают либо как обычные логические операции либо как поразрядные логические операции:

```
AND  A,B       ---> поразрядное логическое И A•B
OR   A,B       ---> поразрядное логическое ИЛИ A+B
XOR  A,B       ---> поразрядное логическое ИСКЛЮЧИТЕЛЬНОЕ ИЛИ A⊕B
NOT  A         ---> поразрядное логическое ОТРИЦАНИЕ ^A
```

Логические значения также возникают в операциях целочисленного сравнения, которые входят в базовый набор слов и имеют общепринятые обозначения:

```
<    A,B       ---> A < B   меньше
=    A,B       ---> A = B   равно
>    A,B       ---> A > B   больше
<=   A,B       ---> A <= B  меньше равно
<>   A,B       ---> A != B  не равно
>=   A,B       ---> A >= B  больше равно
```

Все эти операции сравнения снимают со стека данных два верхних значения, сравнивают их как числа со знаком и возвращают на стек данных результат сравнения в виде значений ИСТИНА или НЕ ИСТИНА в описанном выше представлении.

Логические значения также возникают в операциях сравнения чисел на математическом стеке. Однако, эти операции снимают два верхних значения с математического стека, сравнивают их как числа со знаком и возвращают результат сравнения на стек данных в виде значений ИСТИНА или НЕ ИСТИНА в описанном выше представлении:

```
F<   F:A,B     ---> F: -
     -         ---> A < B   меньше

F=   F:A,B     ---> F: -
     -         ---> A = B   равно

F>   F:A,B     ---> F: -
     -         ---> A > B   больше

F<=  F:A,B     ---> F: -
     -         ---> A <= B  меньше равно

F<>  F:A,B     ---> F: -
     -         ---> A != B  не равно

F>=  F:A,B     ---> F: -
     -         ---> A >= B  больше равно
```

Приведем пример применения логических вычислений:

```
> 32 124 < .
-1 (OK)
> 2.4e-2 124.904 F< .
-1 (OK)
> 1.23 56.5678 F> 34 35 < AND .
0 (OK)
> TRUE DUP NOT . NOT NOT .
0 -1 (OK)
>
```

К логическим операциям над стеком данных также относятся слова логического сдвига влево LSHIFT и вправо RSHIFT. Слово LSHIFT возвращает на стек значение которое получается из A сдвигом его на B разрядов влево и заполнением нулями освобожденных справа разрядов. Аналогично, слово RSHIFT возвращает на стек значение которое получается из A сдвигом его на B разрядов вправо и заполнением нулями освобожденных разрядов слева:

```
    LSHIFT    A,B   --->       (A << B)  логический сдвиг влево
    RSHIFT    A,B   --->       (A >> B)  логический сдвиг вправо
```

Приведем пример применения операций сдвига:

```
> 1 1 LSHIFT .
2 (OK)
> 2 1 RSHIFT .
1 (OK)
> 1 20 LSHIFT .
1048576 (OK)
> -1048576 20 RSHIFT .
4095 (OK)
>
```

Операции сдвига можно использовать для вычисления битовой маски, которая необходима при установке отдельных битов в поразрядных логических операциях. Приведем пример применения операций сдвига для вычисления битовой маски:

```
> : i>m 1 SWAP LSHIFT ; ( index -- mask )
(OK)
> : SetBit i>m OR ; ( flags_before index -- flags_after )
(OK)
> : ClearBit i>m NOT AND ; ( flags_before index -- flags_after )
(OK)
> : InvertBit i>m XOR ; ( flags_before index -- flags_after )
(OK)
```

```
> 10 i>m 20 i>m . .
1048576 1024 (OK)
> 0 10 SetBit .
1024 (OK)
> 1024 10 ClearBit .
0 (OK)
> 0 10 InvertBit InvertBit .
0 (OK)
>
```

Слово i>m вычисляет битовую маску для соответствующего номера бита. Слова SetBit, ClearBit и InvertBit, осуществляют соответствующую манипуляцию над отдельными битами. Например, 1 VAR? 10 SetBit 1 VAR! устанавливает 10 бит в 1 целочисленной переменной.

Структуры управления

В языке ForthLogic™ есть лишь один тип структуры управления процессом выполнения алгоритма - условный оператор. Все другие типы структур управления, такие как многовариантный выбор, условные циклы, циклы со счетчиком и тому подобное можно построить с помощью условного оператора и концепции многозадачности на основе таймеров (см. параграф "Таймеры и многозадачность"). Такой подход гарантирует (особенно для разнообразных безграничных циклов) разделение процессорного времени между отдельными задачами и невозможность его монополизации одной задачей в условиях кооперативной многозадачности в которой форт-система сама функционирует как одна из задач.

Условный оператор строится с помощью слов IF, ELSE и THEN. Эти слова используются лишь внутри определений через двоеточие и разделяют тело определения на отрезки. Фрагмент текста IF <часть-то> ELSE <часть-иначе> THEN задает следующую последовательность действий. Слово IF (если) снимает значение с вершины стека и рассматривает его как логическое. Если это ИСТИНА (любое ненулевое значение), то выполняется фрагмент текста <часть-то> - слова, которые находятся между IF и ELSE, а если НЕ ИСТИНА (ровно нулю), то выполняется фрагмент текста <часть-иначе> - слова между ELSE и THEN. Сами слова ELSE (иначе) и THEN (то) играют роль ограничителей для слова IF и самостоятельного значения не имеют.

Другая форма условного оператора заключается в том, что <часть-иначе> вместе со словом ELSE может отсутствовать и тогда условный оператор имеет сокращенную форму IF <часть-то> THEN. Если логическое значение, которое снимается со стека словом IF есть ИСТИНА, то выполняются слова, которые составляют <часть-то>, а если НЕ ИСТИНА то данный оператор не выполняет никаких действий вообще. Обратите внимание, что в обоих случаях условие для слова IF на вершине стека вычисляется предыдущими словами.

Приведем например определения полезного слова ?DUP, которое дублирует верхнее значение стека если это не нуль, и оставляет стек в предыдущем состоянии если на вершине нуль:

```
> : ?DUP DUP IF DUP THEN ;
 (OK)
>
```

Спецификация данного слова может иметь такой вид (косая черта разделяет два варианта результата, который это слово может оставить на стеке):

```
    ?DUP      A         --->      A, A (A не равно нулю) / A
```

Приведем например определения другого полезного слова LOGIC, которое превращает верхнее значение стека данных в логическое значение пригодное для последующих логических операций:

```
> : LOGIC IF -1 ELSE 0 THEN ;
(OK)
> 12 NOT . 12 LOGIC NOT . 0 LOGIC NOT .
-13 0 -1 (OK)
> 12 45 < 123 LOGIC AND .
-1 (OK)
>
```

Спецификация данного слова может иметь такой вид:

```
    LOGIC     A         --->      -1 (A не равно нулю) / 0
```

Рассмотрим пример построения многовариантного выбора типа SWITCH-CASE на основе условного оператора:

```
> : ТЕМПЕРАТУРА ( температура -- )
DUP 18 < IF ." Умеренно " ELSE
DUP 21 < IF ." Нормально " ELSE
DUP 24 < IF ." Уютно " ELSE
DUP 27 < IF ." Жарко " ELSE
DUP 40 < IF ." Горячо " ELSE
            ." Пожар! "
THEN THEN THEN THEN THEN DROP 1 STRING! ;
(OK)
>
```

В этом примере в зависимости от значения на вершине стека в строчную переменную 1 записывается соответствующее слово. Обратите внимание на одинаковое количество слов DUP, IF, ELSE и THEN.

Преимущество применения условного оператора для построения многовариантного выбора заключается в том, что сравнение и выбор не ограничено лишь целыми числами - можно сравнивать математические и логические величины, можно даже сравнивать разные типы данных в пределах одной такой составленной конструкции.

Таймеры и многозадачность

Как уже упоминалось раньше, язык ForthLogic™ позволяет описывать параллельно выполняемые процессы а сама форт-система функционирует в многозадачной среде. Такое его свойство тесно связано с понятием *таймер.* Таймер - это программный объект который позволяет реализовать

временной интервал по окончании которого может быть выполнено любое известное форт-системе слово.

Пользователю доступны *T_MAX* независимых таймеров, имена которых - это номера в диапазоне от 1 до T_MAX. Количество таймеров T_MAX зависит от аппаратной платформы на которой функционирует форт-система и приведено в применении. Для конфигурации таймеров предназначено слово TIMER!. Слово TIMER! работает следующим образом: с математического стека снимается верхнее значение - временная задержка в секундах, со стека данных снимается верхнее значение - номер таймера, из входного текста выбирается очередное слово (введенное после слова TIMER!) и запускается таймер с соответствующим номером. Смысл выполнения слова TIMER! заключается в следующем: по окончании интервала времени, который равняется значению временной задержки, выполнить слово, которое было указано во время запуска таймера (введенное после слова TIMER!).

Дискретность установки времени задержки срабатывания таймеров составляет 0,01 сек. Внутренняя разрядная сетка представления времени позволяет реализовать временные интервалы от 0 сек. до 21474836,47 сек. с шагом 0,01 сек. То есть, практически реально запустить таймер на 248 дней! Особенный случай времени задержки срабатывания - это 0 сек. (записывается как 0.0). Задержка 0 сек. означает выполнение назначенного данному таймеру слова в следующем такте работы форт-системы - фактически *немедленное* выполнение. Дело в том, что форт-система функционирует в условиях кооперативной многозадачности в составе программно-аппаратной системы контролера, в которой существуют другие программные модули и подсистемы. Все они (равно как и форт-система) выполняются по очереди в отведенных для них тактах. Таким образом, когда выполняется любое слово на языке ForthLogic™, то оно выполняется в пределах такта работы форт-системы и упомянутый выше случай с таймером, который имеет задержку 0 сек. означает, что в следующем такте всей системы будет опять запущена форт-система, которая выполнит назначенное таймеру слово. Любая другая временная задержка для таймера означает постановку форт-системы в очередь отложенных заданий и выполнения лишь через определенный промежуток времени, который отвечает временной задержке.

Приведем пример использования таймера:

```
> : вкл_обогрев 1 1 RO! ;
(OK)
> 4.0 1 TIMER! вкл_обогрев
(OK)
>
```

В этом примере, мы определили новое слово вкл_обогрев, которое записывает логическую единицу в релейный выход S1 (переключает контакты реле - об этом немного позже). Дальше мы запустили таймер 1 на время 4 секунды, по окончании которого будет выполнено слово вкл_обогрев (обратите внимание на десятичную точку в представленном времени задержки). Следовательно, контакты реле переключатся через 4 секунды, начиная от момента времени успешной обработки последней строки текстовым интерпретатором форт-системы (после выведения сообщения (OK)).

Слово TIMER! можно применять и внутри определений через двоеточие, тогда появляется возможность создавать последовательность действий, которые выполняются через определенные промежутки времени. Одна из форм такой последовательности - это создание действий, которые циклически выполняются бесконечное количество раз - то есть, в сущности, это механизм создания *параллельно выполняемых процессов*. Приведем пример:

```
> : сирена 1 RO? NOT 1 RO! 2.0 1 TIMER! сирена ;
```

```
(OK)
> сирена
(OK)
>
```

В этом примере, мы определили новое слово "сирена" со следующим содержанием выполнения: на стек кладется логическое состояние релейного выхода S1, осуществляется логическая инверсия вершины стека, после чего значение со стека записывается в релейный выход S1, дальше запускается таймер 1 на время 2 секунды, по окончании которого будет выполнено слово "сирена" - то есть весь процесс выполнения слова "сирена" опять повторится. После определения слова, мы выполнили слово "сирена", таким образом запустив бесконечное количество раз только что описанный процесс и получив непрерывное переключение контактов реле S1.

Как уже упоминалось, реализованные в системе таймеры являются независимыми, следовательно есть возможность запустить до T_MAX параллельных заданий, каждое из которых описывается своим словом. В такой многозадачной системе необходимо каким-то образом администрировать отдельную задачу. Поскольку, механизм многозадачности реализован с помощью таймеров, то управляя таймерами можно управлять задачами. Для этого предназначено слово TIMER? и слово NAME (имя).

Слово TIMER? работает следующим образом: со стека снимается верхнее значение - номер таймера а на стек кладется или адрес слова, которое будет выполнено соответствующим таймером после заданного промежутка времени, или 0, если слово не задано (таймер не определен).

Адрес слова - это адрес слова в словаре и для его интерпретации предназначено слово NAME, которое работает следующим образом: со стека снимается верхнее значение и интерпретируется как адрес, дальше осуществляется поиск слова, в определении которого содержится данный адрес (каждое определенное слово занимает в словаре диапазон адресов) и если слово найдено, то оно печатается в выходном буфере и на терминале, иначе не печатается ничего. С учетом предыдущего примера, приведем протокол следующего диалога:

```
> 1 TIMER? DUP . NAME
1534 сирена (OK)
>
```

Поскольку в конце выполнения слова "сирена", таймер 1 запускается опять, то в *любой* момент времени, во время выполнения слова TIMER?, на стек будет положен адрес слова "сирена" (показанное значение адреса приведено для примера). Обратите внимание, с помощью слов DUP . NAME мы одновременно вывели на терминал адрес и название слова.

В процессе работы таймера (когда длится обратной отсчет времени) его можно перезапустить на выполнение другого слова через иной промежуток времени. При этом предыдущее слово и предыдущий промежуток времени аннулируются. В частности, если перезапустить таймер со специальным словом STOP (остановить), которое не выполняет ни одного действия, можно фактически остановить задачу которая циклический запускается данным таймером. С учетом предыдущих примеров, приведем протокол следующего диалога:

```
> 0.0 1 TIMER! STOP
(OK)
>
```

Переопределив слово 1 таймера со слова "сирена" на слово STOP, мы остановили бесконечное действие слова "сирена". Задержка 0 секунд, которую мы указали в примере, в данном случае означает немедленное выполнение слова STOP (в описанном выше понимании), однако, может быть произвольной - ведь предыдущее слово и предыдущий промежуток времени 1 таймера уже аннулированы.

Как уже упоминалось, для построения некоторых типов структур управления можно использовать условный оператор и таймеры. В частности, это касается разнообразных условных циклов и циклов со счетчиком. Рассмотрим пример построения на основе условного оператора условного цикла типа BEGIN-UNTIL который существует в языке Форт, однако отсутствует в языке ForthLogic™. Формально, синтаксис построения такого цикла может выглядеть так:

BEGIN <тело-цикла> UNTIL

Такой цикл называется условным циклом с проверкой в конце. После выполнения слов, которые образуют его тело цикла, на стеке остается логическое значение - условие завершения цикла. Слово UNTIL (пока не) снимает это значение и анализирует его. Если это ИСТИНА, то выполнение цикла прекращается. Слово UNTIL можно заменить условным оператором и таймером, который с приемлемой для данной задачи периодичностью будет запускать на выполнения слова, которые образуют тело цикла:

```
> : <тело-цикла> 1 RO? NOT 1 RO! 1 DI? ; ( ---> 1DI )
(OK)
> : BEGIN <тело-цикла> NOT IF 0.1 1 TIMER! BEGIN THEN ;
(OK)
> BEGIN
(OK)
>
```

В этом примере условного цикла слово <тело-цикла> оставляет на стеке логическое значение, которое анализируется в слове BEGIN и если это НЕ ИСТИНА, то слово BEGIN запускается на выполнение через 0,1 сек., иначе слово BEGIN не запускается и действие условного цикла "прекращается". Таким образом, слово UNTIL было заменено рядом слов: NOT IF 0.1 1 TIMER! BEGIN THEN.

Рассмотрим пример построения на основе условного оператора другого условного цикла - цикла со счетчиком типа DO-LOOP который также существует в языке Форт, однако отсутствует в языке ForthLogic™. Формально, синтаксис построения такого цикла может выглядеть так:

DO <тело-цикла> LOOP

Цикл начинается со слова DO (делать), которое снимает со стека два значения: начальное значение (на вершине стека) и конечное значение и запоминает их. Текущее значение счетчика устанавливается равным начальному значению и выполняется тело цикла. Слово LOOP увеличивает значение счетчика на 1 и проверяет условие завершения цикла. Условием завершения цикла является превышение конечного значения на 1. Слово LOOP можно заменить операторами увеличения и проверки счетчика и таймером, который с приемлемой для данной задачи периодичностью будет запускать на выполнения слова, которые образуют тело цикла:

```
> : <тело-цикла> 1 RO? NOT 1 RO! ; ( ---> )
(OK)
> : DO <тело-цикла> 1 VAR? 1 + DUP 1 VAR! 2 VAR? <=
IF 0.1 1 TIMER! DO THEN ;
(OK)
> 1 1 VAR! 10 2 VAR! DO
(OK)
>
```

Слово LOOP в этом примере было заменено рядом слов: 1 VAR? 1 + DUP 1 VAR! 2 VAR? <= IF 0.1 1 TIMER! DO THEN. Переменная 1 VAR используется в качестве счетчика - она, при необходимости, доступна также в теле цикла, а переменная 2 VAR хранит максимальное значение счетчика. Во время работы счетчик увеличивается на 1 и проверяется условие завершения цикла. Перед вызовом слова DO в переменную 1 VAR и 2 VAR необходимо записать минимальное и максимальное значение счетчика.

Векторное выполнение

При определении новых слов в языке ForthLogic™ разрешается применять лишь уже известные форт-системе слова. Однако, существует механизм *обратной ссылки*, который фактически позволяет реализовать определение слов после того как они были предварительно применены в определениях других слов. Этот механизм тесно связан с понятием векторного выполнения и основывается на словах EXECUTE и FIND. Слово EXECUTE (выполнить) работает следующим образом: со стека снимается верхнее значение - *исполнительный адрес* любого слова в словаре и данное слово немедленно выполняется так, как будто его выполнили непосредственно введя в форт-систему.

Внимание: следуют быть чрезвычайно осторожными с исполнительными адресами - если число на вершине стека во время выполнения слова EXECUTE не является таким адресом, то последствия выполнения слова EXECUTE будут непредсказуемыми!

Благодаря слову EXECUTE появляется возможность передавать через стек исполнительные адреса однотипных слов и таким образом реализовывать векторное (то есть, зависимое от контекста) выполнение программы.

Узнать исполнительный адрес любого слова в словаре можно с помощью слова FIND (найти). Слово FIND работает следующим образом: из выходного буфера считывается строки символов - слово для которого необходимо найти исполнительный адрес и, если слово известно форт-системе, на стек кладется исполнительный адрес этого слова. В противном случае на стек кладется значение НЕ ИСТИНА (или 0).

Приведем пример применения упомянутых слов для реализации обратной ссылки:

```
> 0 CONSTANT 1beep_adr
(OK)
> : (1beep) 1beep_adr EXECUTE ;
(OK)
> : 2beep 1.0 1000 BEEP 1.0 1 TIMER! (1beep) ;
(OK)
> : 1beep 1.0 2000 BEEP 1.0 1 TIMER! 2beep ;
(OK)
```

```
> ." 1beep " FIND TO 1beep_adr
(OK)
> 1beep
(OK)
>
```

Сначала мы создали константу 1beep_adr для хранения исполнительного адреса (в равной степени мы могли бы применить для этого целочисленную переменную). Дальше мы создали слово-транслятор (1beep) которое просто выполняет адрес 1beep_adr. Это слово уже можно использовать там где планировались использовать еще не определенное слово 1beep - как, например, в слове 2beep. И наконец, мы определили само слово 1beep. Потом мы записали исполнительный адрес слова 1beep в константе 1beep_adr, тем самым "замыкая" бесконечную последовательность выполнения слов **1beep-2beep**-(1beep)-**1beep**...

Программирование аппаратных средств

Для работы с аппаратными средствами платформы, на которой реализована форт-система, в языке ForthLogic™ существует ряд базовых слов, которые можно разделить по соответствующим категориям: входы, выходы, система, голосовые звонки, SMS, и тому подобное. Список слов для работы с аппаратными средствами может постоянно расширяться, отображая возможность конкретной аппаратуры и потребности конечных пользователей.

Входы

Значения с аналоговых и цифровых входов перед тем как попасть в систему фильтруются. Тип и свойства программных аналоговых фильтров зависят от аппаратной платформы. Цифровые входы фильтруются методом накопления активного и пассивного состояния сигнала с последующим сравнением с пороговыми значениями. Параметры данного алгоритма фильтрации позволяют эффективно поглощать переходные процессы длительностью до 40 мс.

Обращение к входам осуществляется через их номера. Максимальное значение соответствующего номера зависит от аппаратной платформы на которой функционирует форт-система и приведено в применении.

Для работы с цифровыми входами существует слово DI?, которое снимает со стека верхнее значение - номер входа в диапазоне от 1 до DI_MAX и кладет на стек логическое состояние соответствующего цифрового входа (DI/AIx, DIx, Dx). Логическое состояние ИСТИНА означает замыкание соответствующего входа на любой из контактов GND (общий). В случае комбинированных входов DI/AIx, если соответствующий вход с помощью внутренней перемычки и настроек системы сконфигурирован как аналоговый, то логическое состояние этого входа всегда будет НЕ ИСТИНА.

Для работы с комбинированными входами DI/AIx, которые перемычкой и настройками системы сконфигурированы как аналоговые, существует слово AI?, которое снимает со стека верхнее значение - номер входа в диапазоне от 1 до AI_MAX и кладет на математический стек состояние соответствующего комбинированного входа. При этом, если в системе вход сконфигурирован как цифровой, то значение отвечает абсолютному состоянию 10-разрядного АЦП, если вход сконфигурирован как вход по току, то значение отвечает току в миллиамперах, и наконец, если вход сконфигурирован как вход по напряжению, то значение отвечает напряжению в вольтах. Приведем пример применения упомянутых слов:

```
> 1 DI? .
```

```
0 (OK)
> 5 DI? .
-1 (OK)
> 1 AI? F. 2 AI? F.
456.000000 3.234136 (OK)
>
```

Для удобства работы с комбинированными входами DI/AIx в полной версии контролера в которой реализовано конфигурационное меню и фиксированный алгоритм работы, существует слово AIS?, которое снимает со стека верхнее значение - номер входа в диапазоне от 1 до AI_MAX и кладет на математический стек состояние соответствующего комбинированного входа. При этом, если в конфигурационном меню вход определен как цифровой, то значение отвечает абсолютному состоянию 10-разрядного АЦП, если вход определенно как вход по току или напряжению, то значение отвечает масштабированной физической величине. Масштабные коэффициенты принимаются равными значениям указанным в соответствующих пунктах конфигурационного меню. Таким образом слово AIS? можно использовать для получения реальных показателей физической величины (например температура) от датчиков с унифицированным аналоговым выходным сигналом.

Для версий контролеров с универсальными цифровыми входами/выходами реализована поддержка считывателей импульсов. Данная функция работает со всеми входами/выходами и настраивается индивидуально для каждого входу/выхода. Если функция счетчика разрешена, то содержимое счетчика импульсов увеличивается на 1 каждый раз при замыкание соответствующего входа на любой из контактов GND (общий). Максимальное значение счетчика импульсов отвечает целому числу со знаком +2147483647, в случае его превышения счет продолжается от числа -2147483648 в сторону увеличения до 0 и дальше опять до +2147483647.

Для поддержки функции считывателей реализовано три слов: COUNTERSTART настраивает и активирует счетчик, COUNTERSTOP останавливает счетчик а COUNTERGET позволяет получить текущее значение счетчика. Слово COUNTERSTART работает следующим образом: со стека данных снимается верхнее значение - номер счетчика от 1 до DI_MAX, с математического стека данных снимается верхнее значение - временная задержка в секундах, из входного текста выбирается очередное слово (введенное после слова COUNTERSTART) и происходит настройка автоматического обратного вызова этого слова в случае если импульсы на данном счетчике не появляются в течение времени большего чем временная задержка. Перед началом выполнения слова автоматического обратного вызова, на стек данных форт-системой будет положено содержимое данного счетчика импульсов а сам счетчик обнулится. Допускается указывать временную задержку равную 0.0 сек., тогда механизм выполнения слова автоматического обратного вызова будет отменен и счетчик будет непрерывно считать входные импульсы. Слово COUNTERSTOP работает следующим образом: со стека данных снимается верхнее значение - номер счетчика от 1 до DI_MAX и происходит остановка работы данного счетчика без его обнуления. Слово COUNTERGET работает следующим образом: со стека данных снимается верхнее значение - номер счетчика от 1 до DI_MAX, на стек данных кладется содержимое данного счетчика импульсов а сам счетчик обнуляется. Работу считывателей необходимо настраивать каждый раз при старте форт-системы.

Дальше приводится пример настройки работы 1 счетчика на подсчет импульсов от контролера приемника купюр торгового автомата. При последовательном вложении купюр в приемник их стоимость в виде импульсов накапливается в 1 целочисленной переменной:

```
> : cb_billaccep 1 VAR? + 1 VAR! ;
(OK)
> 0.2 1 COUNTERSTART cb_billaccep
```

```
(OK)
>
```

Для версий контролеров с универсальными цифровыми входами/выходами реализована поддержка протокола считывателей идентификаторов (карточек) Wiegand-26, широко употребляемого в системах контроля доступа. Существует возможность подключения до трех считывателей карточек непосредственно к универсальным цифровым входам/выходам контролера. Считыватели могут иметь номера от 1 до 3, причем существует следующее соответствие сигналов считывателя и входов/выходов контролера: сигнал "Data0" считывателя N присоединяется к (N*2-1) входу/выходу контролеру а сигнал "Data1" считывателя N - к (N*2) входу/выходу контролеру.

Для поддержки протокола реализовано автоматическое *ядро протокола считывателей Wiegand-26* и два слова: WIEGANDSTART запускает ядро для одного из считывателей, а WIEGANDSTOP его останавливает. Слово WIEGANDSTART работает следующим образом: со стека данных снимается верхнее значение - номер считывателя в диапазоне от 1 до 3, из входного текста выбирается очередное слово (введенное после слова WIEGANDSTART) и происходит настройка автоматического обратного вызова этого слова в случае успешного получения данных из считывателя. Перед началом выполнения слова автоматического обратного вызова, на стек данных форт-системой будет положен номер карты, который был получен от считывателя. Слово WIEGANDSTOP работает следующим образом: со стека данных снимается верхнее значение - номер считывателя в диапазоне от 1 до 3 и происходит остановка работы ядра протокола для данного считывателя. Ядро протокола считывателей Wiegand-26 необходимо настраивать каждый раз при старте форт-системы.

Дальше приводится пример программы простого контролера доступа с возможностью добавлять и изымать пользователей с помощью двух "мастер-карт". В этом примере Data0 считывателя присоединяется к D1 контролера, Data1 считывателя - к D2 контролера а сам контролер и считыватель питаются от общего источника питания:

```
( array to hold user cards
100 ARRAY cards

6524593 CONSTANT mastercard+ ( add user master card
6524595 CONSTANT mastercard- ( delete user master card

( sounds
: 4b 0.15 1250 BEEP ;
: 3b 0.15 1000 BEEP 0.2 10 TIMER! 4b ;
: 2b 0.15 750 BEEP 0.2 10 TIMER! 3b ;
: correct 0.15 500 BEEP 0.2 10 TIMER! 2b ;
: wrong 0.5 100 BEEP ;

( temporary storages
: lastc! 2 VAR! ;
: lastc? 2 VAR? ;
: tempc! 1 VAR! ;
: tempc? 1 VAR? ;
```

```
( door utilities
: closedoor FALSE 3 DO! ;
: opendoor TRUE 3 DO! 3.0 12 TIMER! closedoor ;

( compare card with user array
: compcard
  DUP lastc! FINDINDEX cards
  IF
    correct opendoor
  ELSE
    wrong
  THEN
;

( add card to user array
: addcard
  0 FINDINDEX cards DUP
  IF
    SET cards
  ELSE
    DROP DROP
  THEN
;

( delete card from user array
: remcard
  FINDINDEX cards DUP
  IF
    0 SWAP SET cards
  ELSE
    DROP
  THEN
;

( wiegand callback
: cb
  tempc? mastercard+ =
  IF
    addcard 0 tempc!
  ELSE
    tempc? mastercard- =
    IF
```

```
      remcard 0 tempc!
    ELSE
      DUP DUP mastercard+ = SWAP mastercard- = OR
      IF
        tempc!
      ELSE
        compcard
      THEN
    THEN
  THEN
;

( sound task
: mastersound
  tempc? mastercard+ = IF 0.2 2400 BEEP THEN
  tempc? mastercard- = IF 0.2 400 BEEP THEN
  0.4 11 TIMER! mastersound
;

( main word
: main mastersound 1 WIEGANDSTART cb ;

." main " BOOT
 main
```

Выходы

Обращение к выходам осуществляется через их номера. Максимальное значение соответствующего номера зависит от аппаратной платформы на которой функционирует форт-система и приведено в применении.

Для работы с цифровыми выходами типа "открытый коллектор" существует слово DO?, которое снимает со стека верхнее значение - номер выхода в диапазоне от 1 до DO_MAX и кладет на стек логическое состояние соответствующего цифрового выхода (DOx, Dx). Логическое состояние ИСТИНА означает, что транзистор соответствующего выхода находится в проводящем состоянии.

Чтобы установить цифровой выход в нужное состояние, существует слово DO!, какое снимает со стека два верхних значения - номер выхода в диапазоне от 1 до DO_MAX и состояние выхода. Дальше, рассматривая это состояние как логическое значение, устанавливается соответствующим образом логическое состояние соответствующего цифрового выхода (DOx, Dx). Логическое состояние ИСТИНА означает, что транзистор соответствующего выхода будет установлен в проводящее состояние.

Для работы с релейными выходами существует слово RO?, которое снимает со стека верхнее значение - номер выхода в диапазоне от 1 до RO_MAX и кладет на стек логическое состояние соответствующего релейного выхода Sx. Логическое состояние ИСТИНА означает, что центральный контакт соответствующего релейного выхода замкнут на нормально разомкнутый контакт данного выхода.

Чтобы установить релейный выход в нужное состояние, существует слово RO!, которое снимает со стека два верхних значения - номер выхода в диапазоне от 1 до RO_MAX и состояние выхода. Дальше, рассматривая это состояние как логическое значение, устанавливается соответствующим образом логическое состояние соответствующего релейного выхода Sx. Логическое состояние ИСТИНА означает, что центральный контакт соответствующего релейного выхода будет замкнут на нормально разомкнутый контакт данного выхода.

```
> 1 DO? . 1 1 DO! 1 DO? .
0 -1 (OK)
> 3 RO? .
-1 (OK)
> 3 RO? NOT 3 RO!
(OK)
> 3 RO? .
0 (OK)
>
```

Для версий контролеров с универсальными цифровыми входами/выходами реализована поддержка двух каналов генерирования широтно-импульсного модулируемого сигнала (ШИМ - англ. PWM). Данная функция поддерживается на аппаратном уровне, потому ее реализация возможна лишь для некоторых универсальных цифровых входов/выходов, а именно: 1 канала ШИМ привязан к входу/выходу D5, 2 канал ШИМ привязан к входу/выходу D6. Рабочий цикл сигнала, или коэффициент заполнения, отвечает разомкнутому состоянию выходного транзистора универсального цифрового входа/выхода и является отношением времени разомкнутого состояния к периоду генерирования ШИМ. То есть, при коэффициенте заполнения 0,99, транзистор большую часть времени практически закрыт.

Для работы с ШИМ предназначено два слова PWMSTART и PWMSTOP. Слово PWMSTART работает следующим образом: с математического стека снимается верхнее значение - коэффициент заполнения сигнала ШИМ в диапазоне от 0,001 до 0,999, со стека данных снимается два верхних значения - номер канала ШИМ в диапазоне от 1 до 2 и частота сигнала ШИМ в диапазоне от 100 Гц до 10000 Гц и в заданном канале начинается непрерывная генерация сигнала ШИМ с заданной частотой и заполнением. Слово PWMSTOP работает следующим образом: со стека данных снимается верхнее значение - номер канала ШИМ в диапазоне от 1 до 2 и в заданном канале прекращается непрерывная генерация сигнала ШИМ. Роботу генератора ШИМ необходимо настраивать каждый раз при старте форт-системы.

Приведем пример применения слов PWMSTART и PWMSTOP для генерации сигнала ШИМ частотой 1000 Гц и заполнением 30% в первом канале (D5):

```
> 0.30 1000 1 PWMSTART
(OK)
> 1 PWMSTOP
(OK)
>
```

Последовательный порт RS485

Обмен по последовательному порту RS485 происходит согласно коммуникационному протоколу MODBUS RTU. Устройства подсоединяются к порту RS485 параллельно и образуют сеть. Обмен в сети MODBUS RTU осуществляется между *главным* устройством (MASTER) и *подчиненными* устройствами (SLAVE). При этом процесс обмена может инициировать лишь главное устройство типа MASTER, а подчиненные устройства типа SLAVE могут лишь отвечать на запросы. Контролеры серии ES-ForthLogic™ могут работать *лишь* в режиме главного устройства.

По умолчанию, данные между главным и подчиненными устройствами пересылаются в виде байтов, упакованных в 11-битовые *посылки* на скорости 9600 битов/сек. Посылка начинается со стартового бита (со значением 0), дальше посылается байт с данными (8 битов) а на конце посылается два стоп бита (со значением 1). Отдельная посылка складывается в *пакеты сообщений* с определенной структурой.

Устройство типа MASTER начинает все пакеты данных с адреса устройства типа SLAVE, к которому адресуется пакет. Каждое устройство типа SLAVE должно иметь уникальный адрес в диапазоне от 1 до 247. В случае когда устройство типа SLAVE высылает ответ на запрос, то в поле адреса оно размещает свой собственный сетевой адрес. Это дает возможность устройству типа MASTER определить откуда поступил ответ.

Следующий байт пакета данных, высланного устройством типа MASTER, образует *код поручения* (функцию), которое должно быть выполнено устройством типа SLAVE. В ответ SLAVE высылает пакет с таким же кодом поручения. Контролеры серии ES-ForthLogic™ могут формировать и обслуживать коды следующих поручений:

- 01 (0x01) Read Coils (Чтение статуса дискретных выходов)
- 02 (0x02) Read Discrete Inputs (Чтение состояния дискретных входов)
- 03 (0x03) Read Holding Registers (Чтение статуса регистров)
- 04 (0x04) Read Input Registers (Чтение состояния входных регистров)
- 05 (0x05) Write Single Coil (Запись отдельного дискретного выхода)
- 06 (0x06) Write Single Register (Запись отдельного регистра)
- 15 (0x0F) Write Multiple Coils (Запись нескольких дискретных выходов)
- 16 (0x10) Write Multiple registers (Запись нескольких регистров)

В последующих байтах пакета высылаются или принимаются данные. Количество данных, которые пересылаются, зависит от кода поручения.

После передачи всех данных, к пакету присоединяется два байта с контрольной суммой CRC. Назначением контрольной суммы CRC является исключение разнообразных ошибок, которые могут возникнуть во время пересылки данных как следствие, например, влияния сильной электромагнитной помехи. Контрольную сумму всегда подсчитывает устройство, которое высылает пакет данных. Дальше устройство, которое принимает пакет, еще раз подсчитывает CRC из полученных данных и сравнивает ее с принятым значением. Если обе суммы совпадают, то устройство приступает к обработке поручения которое содержится в пакете, если нет - пакет игнорируется.

Пакеты сообщений с перечисленными выше кодами поручений формируются и обрабатываются контролерами серии ES-ForthLogic™ автоматически в соответствии с кодом поручения. За это отвечает *ядро протокола MODBUS RTU*. При этом, данные которые высылаются или принимаются, располагаются в глобальных целочисленных и глобальной битовых переменных контролера, что так же зависит от кода поручения. Пакеты сообщений могут формироваться и высылаться в двух режимах: *циклическом* с установленным периодом и *одиночном*. В циклическом режиме параллельно могут формироваться до 4 разных пакетов сообщений пронумерованных от 1 до 4 (не путать этот номер с номерами поручений, адресами подчиненных устройств и тому подобное - это внутренний

для контролера логический номер структуры данных, которая описывает все аспекты обмена), причем это могут быть сообщения как для разных устройств так и для одного и того же но с разными кодами поручений. Такое количество пакетов сообщений, которые могут формироваться параллельно, никоим образом не ограничивает количество устройств, которые могут быть адресованы в сети. Применив одиночный режим формирования пакетов сообщений можно каждый раз произвольно выбирать адрес подчиненного устройства при формировании пакета с тем же номером. Для такого режима формирования пакетов очень удобно применять автоматические обратные вызовы слов, которые происходят в конце одиночного обмена - так называемые обратные вызовы (callback).

Для формирования и активации пакетов сообщений существует слово MODBUSSTART (начать обмен согласно протоколу MODBUS). Слово MODBUSSTART употребляется вместе с числовыми параметрами и системными константами CYCLIC_ACCESS (циклический режим обмена), SINGLE_ACCESS (одиночный режим обмена), READ_COILS (чтение статуса дискретных выходов - функция 01), READ_INPUTS (чтение состояния дискретных входов - функция 02), READ_HOLDREGS (чтение статуса регистров - функция 03), READ_INPUTREGS (чтение состояния входных регистров - функция 04), WRITE_COIL (запись отдельного дискретного выхода - функция 05), WRITE_REG (запись отдельного регистра - функция 06), WRITE_COILS (запись нескольких дискретных выходов - функция 15) и WRITE_REGS (запись нескольких регистров - функция 16), и предназначено для формирования и активации пакетов сообщений согласно коммуникационного протокола MODBUS RTU. Поскольку системные константы просто кладут на стек определенные числа, то синтаксис применения слова MODBUSSTART можно представить в виде диаграммы стековой нотации, однако, учитывая большое количество параметров, более наглядно это можно сделать показав порядок применения числовых параметров и системных констант во время вызова данного слова (порядок слов имеет значение и не следуют забывать о пробелах!):

<period> CYCLIC_ACCESS <slave> <addr> <amount> <index> <func> <packet> MODBUSSTART

- формируется и активируется пакет сообщений в циклическом режиме;

SINGLE_ACCESS <slave> <addr> <amount> <index> <func> <packet> MODBUSSTART

- формируется и активируется пакет сообщений в одиночном режиме;

Числовые параметры в угловых скобках являются обязательными:

<period> *- период формирования пакета в секундах, может принимать значение от 0,1 сек. до 600,0 сек. с шагом 0,01 сек;*

<slave> *- сетевой адрес подчиненного устройства, должен быть уникальным в пределах доной сети и может принимать значение от 0 до 247;*

<addr> *- адрес данных подчиненного устройства; данный адрес в зависимости от кода поручения может быть как адресом дискретного типа данных так и адресом регистрового типа данных и может принимать значение от 0 до 65535;*

<amount> *- количество данных подчиненного устройства; количество в зависимости от кода поручения может быть как количеством данных дискретного типа так и количеством данных регистрового типа и может принимать значение от 1 до 64;*

<index> *- номер глобальной целочисленной или битовых переменной, с которого располагаются данные для записи или результаты считывания, количество задействованных переменных равняется параметру <amount>;*

<func> *- код поручения, может принимать значение READ_COILS, READ_INPUTS, READ_HOLDREGS, READ_INPUTREGS, WRITE_COIL, WRITE_REG, WRITE_COILS,*

WRITE_REGS (см. коды поручений);

<packet> *- номер пакета сообщения, может принимать значение от 1 до 4.*

Естественно, вместо конкретных значений числовых параметров могут быть использованы значения переменных или результаты математической операции, нужно лишь правильно расположить их на стеке. В стековой нотации синтаксис слова MODBUSSTART имеет две формы в зависимости от параметра режима доступа, который представлен системной константой CYCLIC_ACCESS и SINGLE_ACCESS (цифры 1 и 2 соответственно):

```
MODBUSSTART 1,slave,addr,amount,index,func,packet    ---> -
                                          F:period   ---> F:-

MODBUSSTART 2,slave,addr,amount,index,func,packet    ---> -
                                          F:-        ---> F:-
```

Протоколом MODBUS определенно два типа данных:

- дискретный тип - 1-битовая величина, которая может принимать значение 0 или 1 и в поручениях используется для описания дискретных входов и выходов (см. коды поручений);
- регистровый тип - 16-битовая величина, которая может принимать значение от 0 до 65535 и в поручениях используется для описания входных и внутренних (holding) регистров.

В тех поручениях, в которых упоминаются регистровые данные, их содержимое отвечает содержимому целочисленных глобальных переменных контролера - это касается поручений как записи так и чтения. То есть, для поручения записи регистров, значения для записи копируются из содержимого соответствующих целочисленных глобальных переменных, а для поручения чтения регистров прочитанные значения записываются в соответствующие целочисленные глобальные переменные. Соответствие между регистрами подчиненного устройства и целочисленными глобальными переменными контролера устанавливается с помощью параметров <addr>, <amount> и <index>. То же касается и дискретного типа данных, только ему в соответствие ставится 1-битовая глобальная переменная контролера.

Для настройки автоматического обратного вызова слов в одиночном режиме обмена существует слово MODBUSCALLBACK (обратной вызов протокола MODBUS). Слово MODBUSCALLBACK работает следующим образом: со стека данных снимается верхнее значение - номер пакета сообщения, из входного текста выбирается очередное слово (введенное после слова MODBUSCALLBACK) и происходит настройка автоматического обратного вызова этого слова в конце одиночного обмена. Выполнение данного слова является "одноразовым", то есть в конце следующего одиночного обмена оно не будет выполнено если не будет новой настройки с помощью слова MODBUSCALLBACK. Покажем способ вызова данного слова:

<packet> MODBUSCALLBACK <word>

- настройка автоматического одноразового обратного вызова в одиночном режиме;

Параметры в угловых скобках являются обязательными:

<packet> *- номер пакета сообщения, может принимать значение от 1 до 4.*

<word> *- известное форт-системе слово, которое будет выполнено в конце одиночного*

обмена.

Для прекращения формирования пакетов сообщений в циклическом режиме существует слово MODBUSSTOP (остановить обмен согласно протоколу MODBUS). Слово MODBUSSTOP употребляется вместе с числовым параметром номер пакета и предназначено для остановки непрерывного процесса формирования этого пакета сообщений в циклическом режиме. Покажем способ вызова данного слова:

<packet> MODBUSSTOP
- прекращается формирование пакета сообщения в циклическом режиме;

Числовой параметр в угловых скобках является обязательным:

<packet> *- номер пакета сообщения, может принимать значение от 1 до 4.*

В стековой нотации синтаксис слова MODBUSSTOP имеет очень простую форму:

```
MODBUSSTOP          packet ---> -
```

Для определения состояния процесса обмена существует слово MODBUSSTATUS? (состояние обмена согласно протоколу MODBUS). Слово MODBUSSTATUS? снимает со стека номер пакета, который может принимать значение от 1 до 4 и возвращает на стек код состояния обмена для данного пакета сообщения. В стековой нотации синтаксис слова MODBUSSTATUS? имеет форму:

```
MODBUSSTATUS?       packet ---> code
```

Коды состояния обмена разделяются на *коды ошибки главного устройства,* которые возникают в главном устройстве (то есть контролере) и *коды ошибки подчиненного устройства*, которые приходят в ответе на пакет сообщения. Данные коды и их описания представлены в двух следующих таблицах:

Код	Описание кодов ошибки главного устройства
0	Нет ошибки обмена с подчиненным устройством
16	Подчиненное устройство не ответило на поручение в течение заданного интервала времени
17	В ответе подчиненного устройства содержится ошибка контрольной суммы CRC
18	В ответе подчиненного устройства содержится ошибка формата сообщения
19	Внутренняя ошибка главного устройства

Код	Ошибка протокола MODBUS	Описание кодов ошибки подчиненного устройства
1	ILLEGAL FUNCTION	Принятый код поручения не может быть обработан подчиненным устройством.
2	ILLEGAL DATA ADDRESS	Адрес данных указанный в пакете сообщения является не допустимым для данного подчиненного устройства.
3	ILLEGAL DATA VALUE	Величина, которая содержится в поле данных пакета сообщения, является не допустимой величиной для данного

		подчиненного устройства.
4	SLAVE DEVICE FAILURE	Произошла не восстанавливаемая ошибка когда подчиненное устройство пыталось выполнить поручение.
5	ACKNOWLEDGE	Подчиненное устройство приняло поручение и обрабатывает его, но это требует много времени.
6	SLAVE DEVICE BUSY	Подчиненное устройство занято обработкой поручения. Главное устройство должно повторить сообщение позже, когда подчиненное устройство освободится.

Если обмен для данного пакета происходит в циклическом режиме, то код состояния свидетельствует о возможной ошибке обмена главного устройства или, в случае кодов ошибки подчиненного устройства, о том что обмен *прекратился вообще*. Если обмен для данного пакета происходит в одиночном режиме, то код состояния свидетельствует о возможной ошибке *после* того как обмен завершился (при условии, что запрос состояния осуществлен уже после обмена).

Код ошибки 16 возникает когда подчиненное устройство не ответило на поручение в течение заданного интервала времени TIMEOUT. Данный интервал времени по умолчанию равняется 1,0 сек. Однако с помощью слова MODBUSTIMEOUT! (установить время TIMEOUT протокола MODBUS) это время можно изменить. Слово MODBUSTIMEOUT! работает следующим образом: с вершины математического стека снимается число - значение времени в диапазоне от 0,1 сек. до 60,0 сек. с шагом 0,01 сек., и время TIMEOUT устанавливается равным данному значению. В стековой нотации синтаксис слова MODBUSTIMEOUT! имеет следующую форму:

```
MODBUSTIMEOUT!    F:timeout ---> F: -
```

В некоторых случаях существует необходимость изменить параметры обмена по последовательному каналу с принятых по умолчанию на другие (иногда даже нестандартные). Для этого существует слово MODBUSPARAM (установить параметры последовательного порта протокола MODBUS) которое употребляется вместе с числовыми параметрами и системными константами NONE (ни один), EVEN (парный), ODD (непарный) и предназначено для изменения параметров обмена. Покажем способ вызова данного слова:

<parity> <stopbits> <baudrate> MODBUSPARAM

- устанавливает параметры обмена;

Числовые параметры в угловых скобках являются обязательными:

<baudrate> *- устанавливает скорость обмена, может принимать значение из ряда 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200*

<parity> *- значение бита четности, может принимать значение NONE (ни один), EVEN (парный) или ODD (непарный).*

<stopbits> *- количество стоп битов, может принимать значение 1 или 2. При этом 1 стоп бит имеет смысл только при четности NONE. При других типах четности данный параметр значения не имеет.*

В стековой нотации синтаксис слова MODBUSPARAM имеет следующую форму:

```
MODBUSPARAM        parity, stopbits, baudrate ---> -
```

При возникновении нештатных ситуаций во время работы с посторонними устройствами согласно протоколу MODBUS RTU, может существовать необходимость перезапустить ядро протокола MODBUS RTU. Это можно осуществить словом MODBUSRESET (перезапуск ядра MODBUS).

При работе с поручениями, в которых упоминается дискретный тип данных, никаких проблем с интерпретацией данных не возникает: 0 дискретного типа отвечает значению НЕ ИСТИНА битовых глобальных переменных, а 1 дискретного типа отвечает значению ИСТИНА битовых глобальных переменных. При интерпретации значений регистрового типа существует определенная особенность, которая выплывает из того, что регистровый тип является без знаковым. Это значит, что при работе с поручениями, в которых упоминается чтение регистрового типа данных, которые подчиненное устройство трактует как знаковые, необходимо превращать полученные без знаковые числа в знаковые для разрядной сетки целых чисел контролера. Однако при работе с поручениями, в которых упоминается запись регистрового типа данных, необходимость в превращении знаковых чисел в без знаковые отпадает из-за того, что разрядная сетка целых чисел контролера перекрывает как знаковые так и без знаковые числа регистрового типа данных.

Превращение знаковых чисел регистрового типа данных (представленных как без знаковые) в знаковые числа целого типа (принятого в контролере) можно осуществить с помощью слова US>S (без знаковый формат в знаковый). Синтаксис данного слова лучше всего объясняет стековая нотация:

```
US>S    без_знаковое_число ---> число_со_знаком
```

Приведем пример:

```
> 65535 US>S .
-1 (OK)
>
```

Число 65535 попадает в разрядную сетку регистрового типа данных, но в зависимости от интерпретации может значить или 65535 для без знаковых чисел или -1 для знаковых.

Особенность интерпретации также возникает при обмене иными чем регистровый типами данных, например при обмене числами в формате с плавающей запятой одинарной точности (4 байта). Превращение чисел регистрового типа данных в числа математического типа (принятого в контролере) можно осуществить с помощью слова US>F (без знаковый формат в математический). Обратное превращение можно осуществить с помощью слова F>US (математический формат в без знаковый). Некоторые посторонние устройства поддерживают числа математического типа двойной точности (8 байтов). Для их превращения можно воспользоваться словами US>DF (без знаковый формат в математический двойной точности) и DF>US (математический формат двойной точности в без знаковый). Синтаксис данных слов лучше всего объясняет стековая нотация:

```
US>F    число1,число2                --->      -
        F:                      -    --->      F: число

US>DF   число1,число2,число3,число4  --->      -
        F:                      -    --->      F: число
```

```
F>US                -     --->      число1,число2
        F:     число     --->      F: -

DF>US               -     --->      число1,число2,число3,число4
        F:     число     --->      F: -
```

Приведем пример:

```
> 1.2345 F>US . .
1048 16286 (OK)
> 16286 1048 US>F F.
1.234500 (OK)
> -50.9305 DF>US . . . .
0 40960 30490 49225 (OK)
> 49225 30490 40960 0 US>DF F.
-50.930500 (OK)
>
```

Число 1,2345 в двоичном представлении согласно стандарта IEEE-754 одинарной точности имеет вид 0x3F9E0418, в десятичном эквиваленте регистрового типа данных это отвечает числам 16286 (0x3F9E) и 1048 (0x0418). Аналогично превращаются числа двойной точности.

Некоторые приборы нуждаются в передаче по протоколу MODBUS RTU строчных данных. В частности это касается модулей ES-DU-1M собственного производства. Для этого предназначено слово STR>VAR (строки в целочисленную переменную). Слово STR>VAR работает следующим образом: со стека данных снимается верхнее значение - номер переменной а из выходного буфера считывается строки символов, дальше символы начиная с левой стороны строки попарно запоминаются в глобальных целочисленных переменных, начиная с указанного номера. Приведем пример работы с этим словом:

```
> ." hello " 1 STR>VAR 1 VAR? . 2 VAR? . 3 VAR? .
hello 25960 27756 111 (OK)
>
```

Числа 25960, 27756, 111 в шестнадцатеричном представлении имеют вид 0x6568, 0x6c6c, 0x006f - они отвечают кодам символов слова "hello" и расположены так как необходимо для передачи в модуль ES-DU-1M.

Теперь у нас достаточно средств, чтобы запрограммировать работу с подчиненным устройством по протоколу MODBUS (в данном случае со сторонним термопреобразователем AR594 ф. APAR):

```
> : pT 1 VAR? US>S D>F 10.0 F/
." Temp = " F. 1 1 PRINT 1.0 1 TIMER! pT ;
(OK)
> : AR594 1 FPREC! 3.0 1 TIMER! pT
1.0 CYCLIC_ACCESS 1 0 1 1 READ_INPUTREGS 1 MODBUSSTART ;
```

```
(OK)
> AR594
(OK)
>
```

Сначала мы определили слово pT, которое превращает значение температуры, которое находится в 1 переменной, из представления принятого в термопреобразователе в математическое число и печатает его на дисплее (в термопреобразователе температура представлена в виде целого числа со знаком в диапазоне от -1900 до +1900, что отвечает диапазону -190,0° C... +190,0° C). Дальше мы определили слово AR594, которое устанавливает точность отображения чисел, запускает печать температуры и настраивает 1 пакет сообщений на циклическое считывание каждые 1,0 сек. данных регистрового типа из одного входного регистра расположенного по адресу 0 в подчиненном устройстве с логическим адресом 1 и записью этих данных в целочисленную переменную 1.

Приведем также пример обмена с тремя одинаковыми устройствами с использованием одиночного режима обмена и автоматических обратных вызовов:

```
> 0 CONSTANT scan1_a
(OK)
> 0 CONSTANT scan2_a
(OK)
> 0 CONSTANT scan3_a
(OK)
> : (scan1) scan1_a EXECUTE ;
(OK)
> : (scan2) scan2_a EXECUTE ;
(OK)
> : (scan3) scan3_a EXECUTE ;
(OK)
> : scan1 1 MODBUSSTATUS? . SPACE 6 0 PRINT
SINGLE_ACCESS 20 1000 1 1 WRITE_REGS 1 MODBUSSTART
1 MODBUSCALLBACK (scan3) ;
(OK)
> : scan2 1 MODBUSSTATUS? . SPACE 12 0 PRINT
SINGLE_ACCESS 21 1000 1 2 WRITE_REGS 1 MODBUSSTART
1 MODBUSCALLBACK (scan1) ;
(OK)
> : scan3 1 MODBUSSTATUS? . SPACE 1 0 PRINT
SINGLE_ACCESS 22 1000 1 3 WRITE_REGS 1 MODBUSSTART
1 MODBUSCALLBACK (scan2) ;
(OK)
> ." scan1 " FIND TO scan1_a
(OK)
> ." scan2 " FIND TO scan2_a
(OK)
> ." scan3 " FIND TO scan3_a
```

```
(OK)
> : main NONE 1 MODBUSPARAM scan1 ;
(OK)
>
```

Константы scan1_a, scan2_a, scan3_a и слова (scan1), (scan2) (scan3) служат для реализации обратной ссылки на слова scan1, scan2, scan3. В самих же словах scan1, scan2, scan3 реализована печать состояния предыдущего обмена на дисплей, настройка следующего одноразового обмена и настройка автоматического обратного вызова, который, в свою очередь, должен продолжить аналогичный процесс для следующего обмена. Дальше в примере идет связывание обратных ссылок. В слове main происходит настройка параметров обмена и запуск бесконечной цепочки обменов с тремя устройствами. Остановить обмен можно словом MODBUSSTOP:

```
> 1 MODBUSSTOP
(OK)
>
```

Система

Под понятием система, мы имеем в виду обще системные аппаратные и программные ресурсы. Для установки параметров работы системы обычно используется конфигурационное меню (там где оно присутствует) или утилита конфигурации "FLConfig.exe", однако, практически все параметры можно установить программно. На данный момент реализованы следующие слова сгруппированные по определенному признаку, которые предназначены для получения статуса или установки параметров системы. Все настройки сохраняются во внутренней энергонезависимой памяти. Для установки всех параметров в начальное состояние (фабричные настройки) необходимо в терминальном режиме работы указать с новой строки команду RESTORE DEFAULTS.

Идентификация системы

Для более четкой и однозначной идентификации системы, кроме слова VERSION (см. раздел "Введение новых слов"), существует слово SYSINFO. Данное слово печатает в выходной буфер и на терминал расширенную информацию о версии системы, имеющиеся серийные номера и тип лицензии. Имеющиеся серийные номера включают в себя или уникальный серийный номер контролера или IMEI для контролеров со встроенным модулем GSM, а также MAC-адрес для контролеров со встроенным ETHERNET. Тип лицензии показывает какие программные функции ограничены для данного контролера.

Управление системой

Для установки параметров коррекции и опций системных часов существует слово SETWATCH, которое употребляется вместе с системными константами SUMMER_ON и SUMMER_OFF. Вызов данного слова может принимать следующие формы (порядок слов имеет значение!):

<correction> <tzone> SUMMER_ON SETWATCH

- включается автоматический переход на зимнее/летнее время;

Числовые параметры в угловых скобках являются обязательными:

<correction> - *целое число - ежемесячная автоматическая коррекция системных часов в секундах;*

<tzone> - *целое число - часовой пояс в часах*

<correction> <tzone> SUMMER_OFF SETWATCH

- выключается автоматический переход на зимнее/летнее время;

Числовые параметры в угловых скобках являются обязательными:

<correction> - *целое число - ежемесячная автоматическая коррекция системных часов в секундах;*

<tzone> - *целое число - часовой пояс в часах*

Для получения актуальных параметров коррекции и опций системных часов существует слово WATCHPARAM (параметры часов), которое возвращает на стек данные параметры и опции. В стековой нотации синтаксис данного слова имеет следующий вид:

```
    WATCHPARAM     -    ---> <correction>, <tzone>, summer_flag
```

где summer_flag равняется значению ИСТИНА если активирован автоматический переход на зимнее/летнее время.

Для установки системных часов существует слово WATCH!, которое снимает со стека шесть верхних значений календарного времени и устанавливает текущее время системных часов:

<sec> <min> <hour> <year> <mon> <day> WATCH!

- устанавливается текущее время системных часов;

<sec> - *целое число - секунды в диапазоне 0-59.*

<min> - *целое число - минуты в диапазоне 0-59.*

<hour> - *целое число - часы в диапазоне 0-23.*

<year> - *целое число - год в диапазоне 1969-2038.*

<mon> - *целое число - месяц в диапазоне 1-12.*

<day> - *целое число - день месяца в диапазоне 1-31.*

Стековая нотация слова WATCH! имеет следующий вид:

```
    WATCH!    СЕК,МИН,ЧАСЫ,ГОД,МЕС,ДЕНЬ     --->       -
```

Для установки типа комбинированных входов предназначено слово DIAI, которое употребляется вместе с системными константами SET_TO_V, SET_TO_I и SET_TO_D. Дальше представлен порядок слов и констант при применении слова DIAI (порядок слов имеет значение!):

<type> <input> DIAI

- тип комбинированного входа <input> устанавливается равным <type>;

Числовые параметры в угловых скобках являются обязательными:

<type> - тип входа - может принимать значение SET_TO_V, SET_TO_I или SET_TO_D;

<input> - номер входа - целое число в диапазона от 1 до 4.

Для получения актуального типа комбинированных входов существует слово DIAIPARAM (параметр входов), которое возвращает на стек код типа входа. В стековой нотации синтаксис данного слова имеет следующий вид:

```
    DIAIPARAM <input>   --->  <type>
```

Для осуществления процедуры калибровки реализованы следующие слова: CALPOW и CALBAT - калибровка напряжения питания ПЛК; CALV - калибровка комбинированных аналоговых входов в режиме работы по напряжению; CALI - калибровка комбинированных аналоговых входов в режиме работы по току; CALZ - калибровка нулей аналоговых входов. Порядок проведения процедуры калибровки описан в отдельной эксплуатационной документации и предоставляется по требованию.

Дистанционное управление системой с помощью SMS или голосового меню предусматривает осуществление настроек системных опций, которые позволяют определенным образом ограничить права пользователей. Для этого реализован ряд слов и системных констант.

Слово CONTROL (управлять) употребляется вместе с системными константами LOCAL, REMOTE, FOR_LOYAL, FOR_ALL и предназначено для установки опций дистанционного управления. Вызов данного слова может принимать следующие формы (порядок слов имеет значение!):

FOR_ALL REMOTE CONTROL

- дистанционное управление можно осуществлять с любого номера мобильного телефона;

FOR_LOYAL REMOTE CONTROL

- управление можно осуществлять лишь с номеров первых шести избранных пользователей, указанных в конфигурационном меню и с помощью слова USERPHONE (см. дальше);

FOR_ALL LOCAL CONTROL

- дистанционное управление запрещено вообще.

Для получения актуальной опции управления системой существует слово CONTROLPARAM (параметры управления), которое возвращает на стек данную опцию. В стековой нотации синтаксис данного слова имеет следующий вид:

```
    CONTROLPARAM    -    --->  control_all_flag, remote_on_flag
```

где control_all_flag равняется значению ИСТИНА если дистанционное управление можно осуществлять с любого номера; remote_on_flag равняется значению ИСТИНА если дистанционное управление вообще разрешено.

Слово PASSWORD (пароль) употребляется вместе с системными константами PROTECT_BY, DISABLE, строкой-паролем, определенной с помощью слова ." (точка-кавычка) и предназначено для установки опции защиты паролем. Вызов данного слова может принимать следующие формы (порядок слов имеет значение и пароль считывается из выходного буфера!):

PROTECT_BY <pwd> PASSWORD

- включается защита доступа для дистанционного управления, пароль устанавливается равным строке <pwd>;

Строчный параметр в угловых скобках является обязательным:

<pwd> строки пароля в выходном буфере определенная с помощью слова ." (точка- кавычка)

PROTECT_BY PASSWORD

- включается защита доступа для дистанционного управления, пароль не меняется.

DISABLE PASSWORD

- выключается защита доступа для дистанционного управления.

Для получения актуальной опции защиты системы существует слово PROTECTPARAM (параметр защиты), которое возвращает на стек данную опцию. В стековой нотации синтаксис данного слова имеет следующий вид:

```
    PROTECTPARAM   -    --->  protect_on_flag
```

где protect_on_flag равняется значению ИСТИНА если включена защита доступа для дистанционного управления. Другое слово PASSWORD. (пароль-точка) печатает в выходном буфере и на терминале строку актуального пароля.

Для автоматического введения PIN-кода задействованной SIM карты (при активированном PIN-коде), существует слово PIN, которое работает следующим образом: из выходного буфера считывается строки - PIN-код и запоминается во внутренней энергонезависимой памяти. Данный PIN-код автоматически вводится каждый раз при старте системы. Для отображения PIN-кода в выходном буфере и на терминале существует слово PIN. (PIN-точка).

Приведем пример применения слов PIN и PIN.:

```
 > ." 1234 " PIN
 1234 (OK)
 > PIN.
 1234 (OK)
 >
```

Для установки главного слова (или слов) управления системой на языке ForthLogic™ используется слово BOOT (загрузчик), которое считывает из выходного буфера строку текста длиной не более чем 15 символов и запоминает ее в энергонезависимой памяти. В дальнейшем, каждый раз при старте системы (при включении питания или при перезапуске системы) форт-система будет

интерпретировать и выполнять эту строку (для контролера полной версии - при условии установки в конфигурационном меню соответствующей опции). Приведем пример:

```
> : syrena 1 RO? NOT 1 RO! 2.0 1 TIMER! syrena ;
(OK)
> ." 1 2 RO! syrena " BOOT
(OK)
>
```

В этом примере мы определили слово syrena, которое непрерывно переключает контакты 1 релейного выхода, а затем сформировали скрипт управления системой, который состоит из следующих действий: переключить 2 релейный выход и запустить на выполнение слово syrena.

Полезным также может быть слово BOOT. (загрузчик-точка), которое печатает в выходном буфере и на терминале скрипт управления системой. С учетом предыдущего примера:

```
> BOOT.
1 2 RO! syrena (OK)
>
```

Для настройки контролеров, которые содержат встроенный порт ETHERNET, существует ряд слов для установки параметров доступа к локальной сети ETHERNET.

Слова ETHIP, ETHGATEWAY и ETHMASK устанавливают соответственно собственный IP-адрес, IP-адрес основного шлюза и маску подсети. Все эти слова работают следующим образом: из выходного буфера считывается строки - IP-адрес и запоминается в соответствующей ячейке внутренней энергонезависимой памяти. Для отображения собственного IP-адреса, IP-адреса основного шлюза и маски подсети в выходном буфере и на терминале, соответственно существуют слова "с точкой" ETHIP., ETHGATEWAY., ETHMASK.

Приведем пример применения слов ETHIP и ETHIP. (применение остальных слов аналогично):

```
> ." 192.168.2.200 " ETHIP
192.168.2.200 (OK)
> ETHIP.
192.168.2.200 (OK)
>
```

Слово MBTCPPORT! устанавливает порт сервера MODBUS TCP. Слово работает следующим образом: со стека снимается верхнее значение - порт сервера в диапазоне от 1 до 65535 и запоминается в соответствующем ячейке внутренней энергонезависимой памяти. Для получения на стеке актуального значения порта сервера MODBUS TCP существует слово MBTCPPORT?

Приведем пример применения слов MBTCPPORT! и MBTCPPORT?:

```
> 502 MBTCPPORT!
(OK)
> MBTCPPORT? .
502 (OK)
```

>

Регистратор

Встроенный в контролеры серии ES-ForthLogic™ регистратор работает независимо от алгоритма работы контролера и может быть, например, настроен с помощью конфигурационного меню. Однако, для гибкости процесса регистрации, реализован ряд слов и системных констант, которые позволяют полностью осуществить упомянутые настройки непосредственно из программы. В процессе регистрации применяется двойная буферизация на уровне данных и двойная буферизация на уровне файлов данных (о ней детально упомянуто далее по тексту).

Система автоматически и абсолютно прозрачно для пользователя поддерживает механизм двух буферов данных, расположенных в оперативных памяти. Независимо от режима работы, данные сначала попадают в один из этих буферов и при его заполнении автоматически переписываются либо в энергонезависимую память либо передаются через существующее GPRS-соединение, что определяется режимом регистрации. Другой буфер, при этом, позволяет автоматически продолжать регистрацию данных. Такая буферизация является практически необходимой при частой регистрации а также для сохранения ресурса энергонезависимой памяти. Размер буферов данных составляет LOGBUF_MAX байт и позволяет значительно сократить частоту обращения к энергонезависимой памяти, которая имеет ограниченный ресурс (~100000 циклов записи). Параметр LOGBUF_MAX зависит от аппаратной платформы. Применение буферизации данных при регистрации в энергонезависимую память позволяет применять интервалы регистрации от 30 минут и более.

Кроме режимов регистрации в энергонезависимую память через буферы в оперативной памяти, в системе реализован режим регистрации в оперативную память. Такой режим позволяет применять интервалы регистрации от 1 сек до 30 минут и не зависит от ресурса энергонезависимой памяти. При этом размер хранилища в оперативной памяти в байтах равняется двойному значению LOGBUF_MAX.

Слово LOGON (включить регистрацию) употребляется вместе с системными константами EVENTS_MODE (режим событий), USER_MODE (режим пользователя), INTERVAL_MODE (режим интервалов), TO_SD (на SD карту), TO_FLASH (во флешь память), TO_RAM (в оперативную память), TO_TCP (передача по протоколу TCP) ALL_DATA (все данные), INPUTS (входы), OUTPUTS (выходы), и предназначено для запуска процесса регистрации с одновременной установкой опции регистрации. Все настройки немедленно сохраняются во внутренней энергонезависимой памяти. Вызов данного слова может принимать следующие формы (порядок слов имеет значение!):

<data> <dest> EVENTS_MODE LOGON

- запускается процесс регистрации в режиме событий;

<dest> USER_MODE LOGON

- запускается процесс регистрации в режиме пользователя;

<data> <dest> <interv> INTERVAL_MODE LOGON

- запускается процесс регистрации в режиме интервалов;

Числовые параметры в угловых скобках являются обязательными:

<data> *- набор данных, может принимать значение ALL_DATA, INPUTS или OUTPUTS;*

<dest> *- место хранения данных, может принимать значение TO_SD, TO_FLASH, TO_RAM или TO_TCP;*

<interv> *- целое число равно интервалу в секундах.*

Слово LOGON кладет на стек логическое значение - результат своего выполнения. При этом, логическое значение ИСТИНА означает, что регистрация успешно началась, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, процесс регистрации уже длится или не удались открыть файл).

Приведем пример применения слова LOGON:

```
> ALL_DATA TO_FLASH EVENTS_MODE LOGON .
-1 (OK)
> OUTPUTS TO_SD 25 INTERVAL_MODE LOGON .
-1 (OK)
>
```

В первом случае мы запустили "*регистрацию всех данных во флешь-память в режиме событий*". Во втором случае мы запустили "*регистрацию состояния выходов на карту SD/MMC каждые 25 сек. в режиме интервалов*".

Процесс регистрации останавливается словом LOGOFF (прекратить регистрацию), а слово LOGRUN (запустить регистрацию) предназначено для повторного запуска регистрации с уже существующими опциями - оно полезно когда возникает необходимость для частого запуска и остановки уже настроенного процесса регистрации. Слово LOGRUN кладет на стек логическое значение - результат своего выполнения. При этом, логическое значение ИСТИНА означает, что регистрация успешно началась, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, процесс регистрации уже длится или не удались открыть файл).

Для переноса данных регистрации из внутренней энергонезависимой памяти на карту SD/MMC существует слово LOG>SD. Слово LOG>SD работает следующим образом: из выходного буфера считывается строки - название файла и осуществляется попытка либо открыть данный файл на карте SD/MMC, если он существует, либо создать его, дальше происходит перенесение данных из внутренней памяти в данный файл а на стек кладется логическое значение - результат выполнения. При этом, логическое значение ИСТИНА означает, что данные успешно начали переноситься, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, длится процесс регистрации или не удались открыть файл). Допускается не указывать названия файла, тогда будет использован стандартный файл "datalog.txt".

Для передачи данных регистрации из внутренней энергонезависимой памяти по протоколу TCP существует слово LOG>TCP. Слово LOG>TCP осуществляет попытку перенести данные из внутренней памяти через существующее GPRS-соединение согласно протоколу TCP. На стек при этом кладется логическое значение - результат выполнения данного слова: логическое значение ИСТИНА означает, что данные успешно начали переноситься, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, длится процесс регистрации, не удались открыть файл или отсутствует GPRS-соединение).

Передача данных через существующее GPRS-соединение согласно протоколу TCP происходит без дополнительных средств проверки целостности переданных данных - подразумевается, что протокол TCP/IP является достаточно надежным и автоматически обеспечивает целостность данных. Однако, ситуация внезапного разрыва GPRS-соединения может привести к потере данных которые передавались в этот момент. Поэтому нужны дополнительные средства гарантирования успешной доставки данных регистрации. Среди возможных механизмов реализации этого задания была избрана передача данных регистрации на сервер баз данных MYSQL с применением протокола HTTP. Для

этого существуют слова LOG>DB и LOG>SQL. Слово LOG>DB осуществляет попытку перенести данные из внутренней энергонезависимой памяти через существующее GPRS-соединение в режиме CLIENTDB согласно протоколу HTTP. Слово LOG>SQL осуществляет попытку перенести данные из внутренней оперативной памяти через существующее GPRS-соединение в режиме CLIENTSQL согласно протоколу HTTP. На стек при этом кладется логическое значение - результат выполнения данного слова: логическое значение ИСТИНА означает, что данные успешно начали переноситься а НЕ ИСТИНА - что возникли аппаратные проблемы (например, длится процесс регистрации, не удались открыть файл или отсутствует GPRS-соединение в режиме CLIENTDB или в режиме CLIENTSQL).

Передача данных через существующее проводное соединение ETHERNET согласно протокола TCP происходит с помощью слова LOD>DBTCP. Слово LOG>DBTCP осуществляет попытку перенести данные из внутренней оперативной памяти через существующее проводное соединение ETHERNET согласно протокола HTTP. На стек при этом кладется логическое значение - результат выполнения данного слова: логическое значение ИСТИНА означает, что данные успешно начали переноситься а НЕ ИСТИНА - что возникли аппаратные проблемы (например, длится процесс регистрации, не удалось открыть файл или отсутствует проводное соединение ETHERNET с отдаленным сервером).

Процесс переноса данных регистрации из внутренней энергонезависимой памяти может быть достаточно длительным (в зависимости от размера зарегистрированных данных). Для выяснения состояния процесса переноса данных существует слово LOGSEND?, которое кладет на стек логическое значение - состояние процесса переноса данных. При этом, логическое значение ИСТИНА означает, что осуществляется перенос, а НЕ ИСТИНА - что перенос завершен.

Состояние процесса регистрации можно выяснить с помощью слова LOG?, которое кладет на стек логическое значение - состояние процесса регистрации. При этом, логическое значение ИСТИНА означает, что осуществляется регистрация, а НЕ ИСТИНА - что регистрация остановлена.

Для выяснения свободного места во внутренней памяти существует слово FREELOG? (свободное место для регистрации), которое возвращает на стек данных количество свободного места во внутренней энергонезависимой памяти в байтах. Для полной очистки внутренней памяти регистрации существует слово LOGFORMAT (форматировать память регистрации), которое выполняет форматирование внутренней файловой системы с удалением всех зарегистрированных данных.

При регистрирование данных во внутреннюю энергонезависимую память применяется принцип двойной буферизации на уровне файлов данных. Система автоматически и абсолютно прозрачно для пользователя поддерживает систему из двух файлов реестра во внутренней памяти данных. Когда пользователь начинает регистрацию с помощью слов LOGON или LOGRUN то данные попадают в один из файлов реестра (в какой именно - определяет система), который является на данный момент *активным*. Следующая остановка и возобновление регистрации происходят с использованием того же файла. Так происходит до тех пор, пока пользователь не применит одно из слов LOG>SD, LOG>DB или LOG>TCP. С этого момента система автоматически выбирает для регистрирования другой файл реестра, а переписывание данных на карту SD, в базу данных или по протоколу TCP происходит из предыдущего файла реестра. Такой механизм позволяет осуществлять параллельную работу регистратора с переписыванием уже зарегистрированных данных на карту SD, в базу данных или по протоколу TCP. Если процесс переписывания уже зарегистрированных данных заканчивается успешно, то файл с этими данными удаляется из внутренней памяти, если же нет - то он остается и система в следующий раз использует его для дописывания новых данных. Таким образом в системе одновременно могут присутствовать до двух файлов реестра. Это не является проблемой, потому что с помощью двух последовательных процессов переписывания их можно извлечь из внутренней памяти.

При определенных обстоятельствах бывает необходимым вручную изменить активный файл реестра на другой. Для этого предназначено слово LOGFILETOGLE (переключение файла реестра). Также

можно вручную удалить не активный на данный момент файл реестра - для этого предназначено слово LOGFILEDELETE (удаление файла реестра). Если избран режим регистрации во внутреннюю оперативную память, то данное слово удаляет из этой памяти все накопленные данные.

При регистрировании данных во внутреннюю оперативную память двойная буферизация на уровне файлов данных не применяется - есть лишь один файл зарегистрированных данных в оперативной памяти.

Для выяснения наличия зарегистрированных данных во внутренней памяти существует слово LOGFILE? (файлы регистрации), которое кладет на стек числовое значение – количество файлов зарегистрированных данных. При этом, значение 1 или 2 значит, что файлы присутствуют, а значение 0 - что зарегистрированных файлов нет.

Во время автоматической регистрации, данные записываются в виде строк текста в файл "datalog.txt", который размещается либо во внутренней энергонезависимой памяти, либо на карте памяти SD/MMC. Для разных опций регистрации одна запись имеет разный формат и всегда заканчивается символами конца строки и перевода строки - \n\r (в шестнадцатиричном представлении 0x0A и 0x0D):

- ALL_DATA - регистрируется время, дата, напряжения питания, входы и выходы в следующей форме:

"13:04:39|19/03|18.4 13.8|22.23 20.29 0.12 134.34|10000000|0000|000\n\r", где:

13:04:39	19/03	18.4 13.8	22.23 20.29 0.12 134.34	10000000	0000	000
часы, минуты, секунды	день, месяц	напряжение питания, напряжение аккумулятора, В	значение аналоговых входов AI1...AI4	логическое состояние входов DI1...DI8	логическое состояние выходов DO1...DO4	логическое состояние выходов S1...S3

- INPUTS - регистрируется время, дата и входы в следующей форме:

"13:04:39|19/03|22.23 20.29 0.12 134.34 |10000000\n\r", где:

13:04:39	19/03	22.23 20.29 0.12 134.34	10000000
часы, минуты, секунды	день, месяц	значение аналоговых входов AI1...AI4	логическое состояние входов DI1...DI8

- OUTPUTS - регистрируется время, дата и выходы в следующей форме:

"13:04:39|19/03|0000|000\n\r", где:

13:04:39	19/03	0000	000
часы, минуты,	день, месяц	логическое состояние	логическое состояние

секунды		выходов DO1...DO4	выходов S1...S3

Во время автоматической регистрации данных в режиме интервалов происходит периодическая запись строки в указанном выше формате. Во время автоматической регистрации данных в режиме событий запись строки происходит лишь при выявлении любых изменений на унитарных входах или выходах. Дополнительно в этом режиме фиксируются все входные/выходные звонки и SMS в следующем формате:

"13:04:39|19/03|SMS>|+380501234567|Hello world!\n\r", где:

13:04:39	19/03	SMS>	+380501234567	Hello world!
часы, минуты, секунды	день, месяц	тип события	номер телефона	для SMS - текст сообщения

Тип события может быть следующим:

- ">SMS" - поступило входное SMS;
- "SMS>" - отправлено исходящее SMS;
- ">VOICE" - начался входной голосовой вызов;
- "VOICE>" - начался исходящий голосовой вызов;
- "HOLD" - прекратился голосовой вызов.

Значения аналоговых входов, упомянутые выше, представляют собой целые числа, которые отвечают отсчетам 10-разрядного АЦП для соответствующего входа. Однако для полной версии контролера, в которой реализовано конфигурационное меню и фиксированный алгоритм работы, значения отвечают масштабированной физической величине представленной в виде чисел с фиксированной запятой и двумя цифрами после запятой. Масштабные коэффициенты принимаются равными значениям указанным в соответствующих пунктах конфигурационного меню контролера. Таким образом, для полной версии контролера регистрируются реальные показатели физических величин (например температура) от датчиков с унифицированным аналоговым выходным сигналом.

Кроме данных регистрации, которые записываются системой автоматически, существует возможность записать в системный реестр данные непосредственно из программы. Слово LOG (регистрация) записывает содержимое выходного буфера в системный реестр. Данные записываются только во время работы регистратора в режиме пользователя и, по умолчанию, им предшествуют автоматические поля со временем, датой и ключевым словом "ForthLogic>":

"12:32:57|19/03|ForthLogic> [строки текста]\n\r"

Максимальная длина строки текста, которая считывается из выходного буфера, равняется 120 символов. Вышеупомянутые автоматические поля можно с помощью слов LOGTITLEON (включить заголовок регистрации) и LOGTITLEOFF (выключить заголовок регистрации) включать и выключать по желанию в любой момент времени и тем самым освобождать место в выходном буфере для полезных данных. Если возникает задача зарегистрировать строку данных, которая превышает 120 символов,

то это можно осуществить выключив режим автоматического добавления к строке данных символов окончания строки и переноса текста на новую строку "\n\r". В этом случае программа сама определяет когда записать в файл реестра эти символы с помощью слова NEWLINE. Автоматическое добавление символов окончания строки и переноса текста на новую строку можно с помощью слов LOGTERMINON (включить окончание строки) и LOGTERMINOFF (выключить окончание строки) включать и выключать по желанию в любой момент времени.

Приведем пример применения слов для регистрации данных пользователя непосредственно из программы:

```
> : Print IF ." ON " ELSE ." OFF " THEN ;
(OK)
> : log1 ." Bat = " BAT? F. LOG 10.0 3 TIMER! log1 ;
(OK)
> : log2 ." Pump is " 1 RO? Print LOG 15.0 2 TIMER! log2 ;
(OK)
> : StartLog ALL_DATA TO_FLASH USER_MODE LOGON
IF ." Log started " LOG 1 FPREC! 10.0 3 TIMER! log1 15.0 2 TIMER! log2
THEN ;
(OK)
> : StopLog ." Log stoped " LOG
0.0 3 TIMER! STOP 0.0 2 TIMER! STOP ;
(OK)
>
```

Слово Print является вспомогательным: Print печатает в выходном буфере словами "ON" и "OFF" логическое состояние на вершине стекf. Слова log1 и log2 осуществляют, собственно, саму регистрацию данных пользователя через pfвные промежутки времени. Весь механизм запускается и останавливается словами StartLog и StopLog.

Подсистема питания

Слово POW? кладет на математический стек напряжения основного питания в вольтах. Слово BAT? кладет на математический стек напряжения питания аккумулятора в вольтах. Приведем пример применения данных слов:

```
> POW? BAT? F. F.
18.120234 13.780560 (OK)
>
```

Системное время

Слово TIME? кладет на стек три значения в такой последовательности: секунды, минуты, часы системных часов. Соответственно, при снятии со стека, сначала будут полученные часы, потом минуты и, наконец, секунды:

```
TIME?     -     ---> СЕК,МИН,ЧАСЫ
```

Слово DATE? кладет на стек три значения в такой последовательности: год, месяц, день месяца системных часов. Соответственно, при снятии со стека, сначала будет получен день месяца, потом месяц и, наконец, год:

```
     DATE?      -    ---> ГОД,МЕС,ДЕНЬ
```

Слово WDAY? кладет на стек значения дня недели, при этом, воскресенье отвечает значению 0, понедельник - 1, вторник - 2 и так далее:

```
     WDAY?      -    ---> ДЕНЬ_НЕДЕЛИ
```

Слово DST? кладет на стек признак действия летнего времени. Значение ИСТИНА означает действие летнего времени:

```
     DST?       -    ---> dst_flag
```

Слово UTC? кладет на стек состояние системных часов в виде секунд так, как это принято в операционной системе UNIX. Моментом начала отсчета секунд считается полночь с 31 декабря 1969 года на 1 января 1970 года, время с этого момента называют "эрой UNIX" (англ. Unix Epoch). Время UNIX согласовывается с временем UTC. Способ хранения времени в виде количества секунд очень удобно использовать при сравнении дат (с точностью до секунды), а также для хранения дат - при необходимости их можно превратить в любой удобочитаемый формат.

Для превращения общепринятого представления времени в формат UTC существует слово >UTC.

```
     >UTC       СЕК,МИН,ЧАСЫ,ГОД,МЕС,ДЕНЬ    --->     UTC
```

Для превращения формата времени UTC в общепринятое представление времени существует ряд слов: >TIME, >DATE, >WDAY. Эти слова снимают со стека значение времени в виде секунд UTC и возвращают на стек время, дату и день недели. Лучше всего описать работу данных слов с помощью стековой нотации:

```
     >TIME          UTC       --->      СЕК,МИН,ЧАСЫ
     >DATE          UTC       --->      ГОД,МЕС,ДЕНЬ
     >WDAY          UTC       --->      ДЕНЬ_НЕДЕЛИ
```

При этом, дни недели интерпретируются следующим образом: воскресенье отвечает значению 0, понедельник - 1, вторник - 2 и так далее.

Для печати в выходном буфере и на терминале форматированного общепринятого календарного времени существует слово TIMESTAMP (метка времени). Календарное время печатается в следующем виде (без кавычек): "часы:минуты:секунды день/месяц/год". Например "15:02:45 10/01/11".

Для сравнения между собой общепринятых дат и времен существует слово ISNOW, которое попарно сравнивает шесть верхних значений на стеке данных (убирая их со стека) и возвращает на стек логическое значение ИСТИНА только в том случае, если все сравнения истинны, и НЕ ИСТИНА в противном случае. Сравнение происходит в особенном порядке:

```
    ISNOW     A1,B1,C1,C2,B2,A2    --->       (A1=A2)•(B1=B2)•(C1=C2)
```

если A1=A2, B1=B2, C1=C2, то на стеке будет -1 (ИСТИНА), иначе 0 (НЕ ИСТИНА). Приведем пример применения всех вышеупомянутых слов:

```
> DATE? . . . TIME? . . .
21 6 2008 14 7 2 (OK)
> UTC? .
1214057197 (OK)
> TIME? DATE? >UTC .
1214057199 (OK)
> 23 0 0 TIME? ISNOW . 21 6 2008 DATE? ISNOW .
0 -1 (OK)
> 0 0 15 DATE? WATCH!
(OK)
>
```

В последней строке приведен пример установки системного времени равным 15:00:00 без изменения даты.

Для задач управления освещением реализована функция *астрономического таймера*. Данная функция позволяет получить моменты времени восхода и заката солнца. Функция расчета моментов восхода и заката Солнца реализуется для текущей даты на основе показаний системных часов. В качестве параметров используются географические координаты места установки контролера и тип расчета. Географические координаты указываются с учетом знака в пределах от -90.0 до +90.0 градусов для широты (положительная для Северной, негативная для Южной широты) и в пределах от -180.0 до +180.0 градусов для долготы (положительная для Восточной, негативная для Западной долготы). Тип расчета выбирается из двух возможных: гражданские сумерки и рассветы и официальный восход и закат Солнца. Гражданские сумерки - когда верхний край диска Солнца опускается на 6 градусов ниже горизонта. В этом случае деятельность человека нуждается в дополнительном уличном освещении. Гражданский рассвет - когда начинает светать и верхний край диска Солнца еще находится на 6 градусов ниже горизонта. Официальный закат Солнца - момент когда верхний край диска Солнца исчезает за горизонтом. Официальный восход Солнца - момент когда верхний край диска Солнца появляется над горизонтом.

Функция расчета моментов восхода и заката Солнца представлена двумя словами и двумя системными константами. Слова SUNRISE? (восход солнца) SUNSET? (закат солнца) работают следующим образом. С математического стека они снимают два числа - широту и долготу, с обычного стека снимают тип расчета, который задается с помощью системных констант CIVIL (гражданские сумерки и рассветы) и OFFICIAL (официальный восход и закат Солнца), и кладут на стек три значения в такой последовательности: секунды, минуты и часы соответствующего момента восхода или заката солнца. Соответственно, при снятии со стека, сначала будут получены часы, потом минуты и, наконец, секунды. В стековой нотации данные слова имеют следующий вид:

```
    SUNRISE?  тип_расчета          ---> СЕК,МИН,ЧАСЫ
              F: ШИРОТА,ДОЛГОТА    ---> -

    SUNSET?   тип_расчета          ---> СЕК,МИН,ЧАСЫ
```

```
                    F: ШИРОТА,ДОЛГОТА   ---> -
```

Если для данной местности Солнце не может взойти или спрятаться за горизонт, то значение секунды, минуты и часов соответствующего момента восхода или заката солнца будет равняться 0. Приведем пример программы по применению упомянутых слов:

```
  ( Astronomic timer example                               )
  (    ___               __________________                )
  (DO2    \______________/                  \____          )
  (      dawn          dusk                   dawn         )
  (                                                        )
  ( Used timers: #6 - for 0.75 sec offtask cycle           )
  (              #7 - for 0.75 sec ontask cycle            )

49.82766 FCONSTANT Lattitude
24.00867 FCONSTANT Longitude

: sunrise? Lattitude Longitude OFFICIAL SUNRISE? SWAP ROT ;
: sunset? Lattitude Longitude OFFICIAL SUNSET? SWAP ROT ;
: dawn? Lattitude Longitude CIVIL SUNRISE? SWAP ROT ;
: dusk? Lattitude Longitude CIVIL SUNSET? SWAP ROT ;

: offtask dawn? TIME? ISNOW IF 0 2 DO! THEN 0.75 6 TIMER! offtask ;
: ontask dusk? TIME? ISNOW IF 1 2 DO! THEN 0.75 7 TIMER! ontask ;

( main word
: main offtask ontask ;

." main " BOOT
 main
```

Интерфейс пользователя

Клавиатура

Для контролеров со встроенной клавиатурой и дисплеем, для настройки функции клавиш **F1, F2,** △ ▽, ◁ ▷ и **OK** существует слово BUTTON, которое применяется вместе с системными константами имена которых отвечают названию клавиш: F1, F2, UP (△), DOWN (▽), LEFT (◁), RIGHT (▷), OK. Эти константы кладут на стек номера соответствующих клавиш. Слово BUTTON работает следующим образом: со стека снимается верхнее значение - номер клавиши в диапазоне от 1 до 7, из входного текста выбирается очередное слово (введеное после слова BUTTON), которое будет выполняться при нажатии соответствующей клавиши, и осуществляется настройка данной клавиши согласно заданных параметров. Приведем пример применения слова BUTTON для настройки функции клавиши F2:

```
  > : inversija 3 RO? NOT 3 RO! ;
  (OK)
```

```
> F2 BUTTON inversija
(OK)
>
```

В процессе выполнения задачи пользователя, функции клавиш можно многократно изменять в зависимости от контекста задачи. Определенные таким образом функции клавиш **F1** и **F2** будут активными во *всех режимах интерфейса* кроме случая, когда в режиме меню происходит введение значения, а функции клавиш △, ▽, ◁ ▷ и **OK** будут активными лишь в *рабочем режиме интерфейса*.

Слова которые назначены клавишам не запоминаются при выключении питания контролера. Это следует учитывать при программировании задач пользователя.

Дисплей

Выведение информации

Для контролеров с графическим цветным дисплеем, для выведения текстовой информации на дисплей существует слово PRINT (печатать). Выведение возможно лишь в *рабочем режиме интерфейса* - в других режимах форт-система блокирует выведение. Рабочее поле для выведения текстовой информации представляет собой прямоугольный участок посредине дисплея. Максимальное количество строк и символов в строке зависит от аппаратной платформы на которой функционирует форт-система и приведено в приложении. Нумерация строк - сверху вниз от 0 до *ROW_MAX-1*, нумерация символов - слева направо от 0 до *COL_MAX-1*. Текст отображается с помощью большого шрифта в одном из 9 цветов (белый, красный, оранжевый, желтый, зеленый, голубой, синий, фиолетовый, черный) или на белом фоне или на черном. По умолчанию (после подачи питания на контролер), выведение происходит черным шрифтом на белом фоне.

Слово PRINT работает следующим образом: со стека снимается два верхних значения - координаты позиции выведения текста, которые состоят из номера строки и номера символа, из выходного буфера считывается строки - текст, который будет выводиться в рабочее поле, начиная с данной позиции, и происходит выведение на активный фон дисплея в активном цвете. При достижении конца строки, символы текста автоматически переносятся в следующую строку. В последней строке, при достижении конца строки, последующее выведение блокируется.

Активный цвет шрифта можно установить с помощью слов WHITE (белый), RED (красный), ORANGE (оранжевый), YELLOW (желтый), GREEN (зеленый), BLUE (голубой), DEEPBLUE (синий), VIOLET (фиолетовый) и BLACK (черный), которые работают следующим образом: устанавливают соответствующий *активный* цвет текста. Все последующие выведения происходят в активном цвете вплоть до его последующей смены.

Активный фон выведения (а также активный на данный момент цвет выведения) с помощью слова INVERT (инвертировать) можно поменять на противоположный, осуществляя таким образом инверсию цветов, в результате которой белый фон становится черным (или наоборот) а цвет текста меняется на дополняющий ("негативный"). Все последующие выведения происходят в данном цвете на данный фон вплоть до их последующей смены.

Для чистки рабочего поля выведения текстовой информации существует слово CLEAR (очистить), которое вытирает все символы в соответствии с активным фоном - рабочее поле становится или белым или черным.

Приведем пример применения слов для работы с дисплеем:

```
> ." "Красный" " RED 0 0 PRINT
(OK)
```

```
> ." "Зеленый" " GREEN 0 1 PRINT
(OK)
> ." "Инверсия зеленого" " INVERT 0 2 PRINT
(OK)
> ." "Синей на черном" " DEEPBLUE 0 4 PRINT
(OK)
> CLEAR
(OK)
>
```

Слово "Красный" будет выведено, начиная с символа 0 строки 0 в красном цвете. Слово "Зеленый" будет выведено, начиная с символа 0 строки 1 в зеленом цвете. Фраза "Инверсия зеленого" будет выведена, начиная с символа 0 строки 2 в фиолетовом цвете на черном фоне. Фраза "Синий на черном" будет выведена, начиная с символа 0 строки 4 в синем цвете на черном фоне. Обратите внимание, что после выполнения слова CLEAR, рабочее поле (фон) станет черным.

Введение информации

Для введения числовых параметров с помощью клавиш и дисплея существует слово GET, которое работает следующим образом: с математического стека снимается верхнее значение **-** число, которое будет отображено в поле окна для введения значений, из выходного буфера считывается строки **-** текст длиной не больше 12 символов, который будет отображен в шапке окна для введения значений, из входного текста выбирается два очередных слова обратного вызова (введенных после слова GET) и на дисплее отображается стандартное окно для введения значений.

При редактировании значений в окне для введения, перемещение курсора осуществляется клавишами ◁▷, при этом курсор подсвечивается розовым цветом. Клавишей **F1** осуществляется выбор набора символов из списка: [S] **-** знаки пунктуации; [D] – цифры; [L] **-** латинские большие буквы; [l] **-** латинские малые буквы; [К] **-** кириличные большие буквы; [к] **-** кириличные малые буквы; при этом отображение активного набора символов осуществляется в левом верхнем угле окна для введения (на синем фоне). Клавишами △▽осуществляется перебор символов из выбранного набора. Удаление символа в позиции курсора осуществляется клавишей **F2,** утверждение выбранного значения **-** клавишей **OK,** а выход без внесения изменений **-** клавишей **Esc**.

Когда пользователь закончит ввод и нажмет клавишу **OK,** форт-система выполнит первое из указанных слов обратного вызова, причем, на момент выполнения данного слова, на математическом стеке будет находиться введенное значение. В случае, когда пользователь нажмет клавишу **Esc,** форт-система выполнит второе из указанных после слова GET слов обратного вызова. Введение числовых параметров возможно лишь в *рабочем режиме интерфейса* и в пределах меню "Функции Пользователя" **-** в других случаях форт-система блокирует роботу слова GET. Приведем пример применения слова GET:

```
> : print F. 0 0 PRINT ;
(OK)
> ." ТЕМП > " 65.0 GET print STOP
ТЕМП > (OK)
>
```

Для введения строчных параметров с помощью клавиш и дисплея существует слово GETS, которое работает следующим образом: со стека снимается верхнее значение - номер строчной переменной с текстом, который будет отображен в поле окна для введения значений и для сохранения введенной строки, из выходного буфера считывается строки - текст длиной не больше 12 символов, который будет отображен в шапке окна для введения значений. Дальше, из входного текста выбирается два очередных слова обратного вызова (введенных после слова GETS) и на дисплее отображается стандартное окно для введения значений. Правила редактирования значений аналогичны с описанными в предыдущем параграфе для слова GET.

Когда пользователь закончит ввод и нажмет клавишу **OK,** форт-система выполнит первое из указанных слов обратного вызова, причем, на момент выполнения данного слова, в указанной строчной переменной будет находиться введенная строки. В случае, когда пользователь нажмет клавишу **Esc,** форт-система выполнит второе из указанных после слова GETS слов обратного вызова. Введение строчных параметров также возможно лишь в *рабочем режиме интерфейса* и в пределах меню "Функции Пользователя" - в других случаях форт-система блокирует роботу слова GETS. Приведем пример применения слова GETS:

```
> : print 1 STRING? 0 0 PRINT ;
(OK)
> ." Timout " 1 STRING!
(OK)
> ." СЛОВО 1 > " 1 GETS print STOP
СЛОВО 1 > (OK)
>
```

Меню пользователя

Для контролеров с графическим дисплеем можно настраивать особенное меню пользователя, которое функционирует в составе обще системного конфигурационного меню. Максимальное количество пунктов такого меню зависит от аппаратной платформы на которой функционирует форт-система и приведено в приложении. Для настройки пунктов меню пользователя, существуют слова MENU и HIDE. Слово MENU работает следующим образом: со стека снимается верхнее значение - номер меню в диапазоне от 0 до MENU_MAX, из выходного буфера считывается строки - текст, который будет отображаться в соответствующем пункте меню, из входного текста выбирается очередное слово (введенное после слова MENU), которое будет выполняться при выборе соответствующего пункта меню. Дальше осуществляется настройка соответствующего пункта меню согласно заданных параметров и данный пункт меню становится видимым. Допустимо не указывать текстовую строку для отображения, тогда название пункта меню останется без изменений. Номера меню от 1 до MENU_MAX отвечают пунктам меню пользователя, а номер меню 0 отвечает за настройку функции, которая будет выполняться каждый раз при вхождении к меню "Функции Пользователя". Такая функция является полезной для начальной инициализации структуры меню пользователя. Нужно также помнить, что максимальная длина строки, которая будет отображаться в названии пункта меню пользователя, составляет 15 символов. Причем, для пункта меню с номером 0, текст вообще не имеет значения (игнорируется). Приведем простой пример применения данного слова:

```
> : inversija 3 RO? NOT 3 RO! ;
(OK)
> ." Инверсия " 1 MENU inversija
Инверсия (OK)
```

```
>
```

В этом примере мы определили новое слово inversija которое выполняет следующие действия: на стек кладется логическое состояние релейного выхода S3, осуществляется логическая инверсия вершины стека, после чего значение со стека записывается в релейный выход S3. Дальше мы назначили это слово первому пункту меню пользователя, определив при этом название пункта как "Инверсия".

Слова, которые назначены пунктам меню а также текст, который отображается в пунктах меню пользователя не запоминаются при выключении питания контролера. Это следует учитывать при программировании задач пользователя.

Слово HIDE работает следующим образом: со стека снимается верхнее значение - номер меню в диапазоне от 1 до MENU_MAX, дальше осуществляется выключение соответствующего пункта меню - оно становится невидимым и недоступным для пользователя. Повторное включение такого пункта меню происходит словом MENU. Таким образом можно реализовать динамическое меню пользователя, пункты которого зависят от контекста задачи.

Определить, какой пункт был выполнен последним можно с помощью слова LASTMENU?, которое кладет на стек номер пункта меню пользователя, который был выполнен последним. Данное слово позволяет персонализировать работу с отдельными пунктами меню.

При каждом выполнении слов MENU и HIDE, *фокус меню* - то есть выделенная видимая строки остается без изменений. Для большей гибкости при созданные многоуровневых меню, существует слово FOCUS которое работает следующим образом: со стека снимается верхнее значение - номер меню в диапазоне от 1 до MENU_MAX и осуществляется установка фокуса на данный пункт меню - то есть он становится выделенным розовым цветом и отображается первой строкой в существующей структуре меню.

Дополнительно, для большего удобства использования меню, существует возможность выведения вспомогательной текстовой информации в информационном поле меню пользователя (левая верхняя часть дисплея). Для этого существует слово INFO (информация), которое из выходного буфера считывает строку текста длиной до 15 символов и отображает его в левой части информационного поля. Данное слово работает лишь в *пределах меню пользователя*.

Все перечислены выше слова для работы с меню пользователя позволяют создавать многоуровневые динамические меню пользователя, которые не уступают по возможностям встроенному конфигурационному меню.

Звук

Для формирования звукового сигнала с помощью встроенного *генератора звуковых сигналов*, существует слово BEEP (сигнал), которое работает следующим образом: с математического стека снимается верхнее значение - длительность сигнала в диапазоне (0,001 ... 64,0) секунд с шагом 0,001 сек., со стека данных снимается верхнее значение - частота звукового сигнала в диапазоне (100 ... 10000) Гц и происходит генерация одиночного звукового сигнала с заданной частотой и длительностью.

Слово BEEP позволяет дополнительно оформить интерфейс с пользователем звуковыми событиями, что, в целом, увеличивает удобство данного интерфейса. Приведем пример применения слова BEEP для генерации сигнала частотой 1000 Гц и длительностью 2 секунды:

```
> 2.0 1000 BEEP
(OK)
>
```

Индикация

Для индикации с помощью встроенных светодиодов, существуют слова LED? и LED!, которые используются вместе с системными константами F1 и F2. Имена этих констант отвечают названиям светодиодов и они кладут на стек номера соответствующих светодиодов. Слово LED? работает следующим образом. Со стека данных снимается верхнее значение - номер светодиода, а на стек данных кладется состояние соответствующего светодиода, которое принимает значение ИСТИНА или НЕ ИСТИНА. Слово LED! работает иным образом. Со стека данных снимается два верхних значения - номер светодиода и новое значение состояния светодиода в виде ИСТИНА или НЕ ИСТИНА. Состояние ИСТИНА означает что светодиод светится. Приведем пример работы с этими словами:

```
> TRUE F1 LED!
(OK)
> F1 LED? . F2 LED? .
-1 0 (OK)
>
```

Акустическая система

Некоторые модели контролеров оборудованы акустической системой (далее по тексту - *аудиосистема*), через которую можно воспроизводить сообщения и числа аналогично тому как это делается при воспроизведении голосового меню через GSM модуль.

Для воспроизведения звукового сообщения через аудиосистему в любой момент времени, существует слово AUDIOPLAY (воспроизвести аудио) которое работает следующим образом: из выходного буфера считывается строки - название файла в формате *.wav, который расположен на карте памяти SD/MMC (процедура создания звуковых файлов описана в приложении), из входного текста выбирается очередное слово обратного вызова (введенное после слова AUDIOPLAY) и запускается процесс воспроизведения звукового сообщения, а на стек кладется логическое значение - результат выполнения слова AUDIOPLAY. При этом, логическое значение ИСТИНА означает, что процесс воспроизведения успешно начался, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, файл или карта памяти отсутствуют или уже осуществляется воспроизведение через аудиосистему и тому подобное). Когда звуковое сообщение будет полностью воспроизведено, форт-система выполнит указанное после слова AUDIOPLAY слово обратного вызова.

Для воспроизведения через аудиосистему словами значения числа с вершины математического стека существует слово AUDIOSAY (сказать) которое работает следующим образом: из выходного буфера считывается строки - путь к набору файлов чисел в формате *.wav, который расположен на карте памяти SD/MMC, с математического стека снимается верхнее значение, которое должно находиться в пределах от -999,9 до +999,9 включительно, из входного текста выбирается очередное слово обратного вызова (введенное после слова AUDIOSAY) и запускается процесс воспроизведения словами значения числа, а на стек данных кладется логическое значение - результат выполнения слова AUDIOSAY. При этом, логическое значение ИСТИНА означает, что процесс воспроизведения успешно начался, а НЕ ИСТИНА - что возникли аппаратные проблемы или число находится вне установленных пределов. Когда значение числа будет полностью воспроизведено, форт-система выполнит указанное после слова AUDIOSAY слово обратного вызова. Набор файлов чисел в формате *.wav описан в приложении.

По умолчанию, значение числа с вершины математического стека воспроизводится с указыванием знака числа и с точностью 1 цифра после десятичной запятой. Воспроизведение знака числа можно отключить с помощью слова NOAUTOSAYPLUS и повторно включить с помощью слова AUTOSAYPLUS.

Для прекращения воспроизведения звукового сообщения через аудиосистему или числа с вершины математического стека, существует слово AUDIOMUTE, которое немедленно прекращает воспроизведение звука через аудиосистему и блокирует выполнение заданного после слов AUDIOPLAY или AUDIOSAY слова обратного вызова.

Модуль GSM/GPRS

Состояние сети GSM

Для обеспечения гибкости и удобства использования GSM модуля, в форт-системе доступна информация, которая касается состояния сети GSM, состояния связи и тому подобное. На данный момент реализованы следующие слова для получения дополнительной информации о состоянии модуля и сети.

Слово ROAMING? возвращает на стек логическое значение ИСТИНА, если GSM модуль зарегистрировался в роуминге или логическое значение НЕ ИСТИНА если GSM модуль зарегистрировался в домашней сети. Данное слово может пригодиться в пограничных местностях для программного ограничения трафика в случае регистрации в роуминге.

Слово OPERATOR печатает в выходном буфере и на терминале название оператора сети GSM, в которой зарегистрирован GSM модуль. Если модуль не зарегистрирован, то строки будет пустой.

Слово SIGNAL? возвращает на стек числовое значение уровня сигнала сети GSM. Качество сигнала при успешной регистрации в сети, может принимать значение от 0 до 4. Если возникли проблемы с регистрацией или самим GSM модулем то слово SIGNAL? возвращает на стек число -1. Данное слово может служить индикатором начала успешной работы в сети GSM. Информация о качестве сигнала в сети GSM обновляется форт-системой каждые 5 сек.

Слово SIM? возвращает на стек логическое значение ИСТИНА, если SIM-карта установлена в контролере и успешно активирована или логическое значение НЕ ИСТИНА в противоположном случае.

Для получения более подробной информации о качестве сигнала в сети GSM и состоянии GSM модуля существует слово MODULE? которое возвращает на стек четыре значения:

```
    MODULE?   -    --->       rssi,ber,voltage,temp
```

где *rssi* - внутреннее представление качества сигнала сети GSM, rssi может принимать значение от -1 до 31, значение -1 свидетельствует о невозможности обнаружить сигнал, приблизительное соотношение между значением rssi и мощностью сигнала в dBm:

0	-115 dBm или меньше
1	-111 dBm
2...30	-110...-54 dBm
31	-52 dBm или больше;

ber - процент ошибки при передаче данных GPRS;

voltage - внутреннее напряжение питания GSM модуля в мВ;

temp - внутренняя температура GSM модуля в градусах.

Приведем пример применения данных слов:

```
 > ROAMING? .
 0 (OK)
```

```
> OPERATOR
Mobile GSM (OK)
> SIGNAL? .
3 (OK)
>
```

Номера телефонов

В последующих примерах все номера телефонов могут быть статически определены с помощью строк. Для того, чтобы иметь возможность менять номера телефонов без перестройки всей задачи, в энергонезависимой памяти существуют специальные строчные переменные длиной 15 символов для хранения номеров телефонов пользователей. Для работы с данными переменными введены слова USER (пользователь), которое позволяет использовать при описании задачи телефонные номера пользователей - фактически считывать эти специальные переменные, и слово USERPHONE (телефон пользователя) которое позволяет определять и изменять телефонные номера пользователей - то есть записывать новые значения в эти специальные переменные. Максимальное количество специальных строчных переменных для хранения номеров телефонов зависит от аппаратной платформы на которой функционирует форт-система и приведено в приложении. Причем, первые шесть переменных имеют статус переменных для хранения телефонных номеров *избранных пользователей*, которые могут устанавливаться с помощью конфигурационного меню и принимают участие в политике безопасности всей системы - при определенных условиях лишь с этих телефонов можно осуществлять дистанционное управление.

Слово USER работает следующим образом: со стека снимается верхнее значение, которое может находиться в диапазоне от 1 до PHONE_MAX - это порядковый номер телефона в списке (или номер пользователя), дальше в выходном буфере и на терминале печатается номер телефона, который хранится в специальной строчной переменной с этим номером.

Слово USERPHONE работает следующим образом: со стека снимается верхнее значение, которое может находиться в диапазоне от 1 до PHONE_MAX - это порядковый номер телефона в списке (или номер пользователя), из выходного буфера считывается строки - номер телефона в международном или национальном формате и запоминается в специальной строчной переменной с этим номером во внутренней энергонезависимой памяти. Приведем пример применения слов USER и USERPHONE:

```
> ." +380507777777 " 1 USERPHONE
+380507777777 (OK)
> 1 USER
+380507777777 (OK)
>
```

Также существует слово LAST (последний), которое позволяет использовать при описании задачи номер телефона последнего абонента. Слово LAST печатает в выходном буфере и на терминале номер телефона абонента, который последним дозвонился на контролер, или послал последнее SMS сообщение.

Приведем еще примеры применения данных слов:

```
> 3 USER LAST
0501234567 +380507654321 (OK)
>
```

SMS и USSD

Выходные SMS представляют один из способов сообщить пользователю о тех или других событиях. Однако, входные SMS представляют собой не просто возможный ответ пользователя, а намного более мощную концепцию реализованную с помощью языка ForthLogic™. Дело в том, что все входные SMS непосредственно попадают (за небольшим исключением) во входной буфер текстового интерпретатора форт-системы и, соответственно, интерпретируются и выполняются. Данная концепция чрезвычайно гибкая и мощная, однако, с целью повышения надежности и безопасности целой форт-системы, во входных SMS *заблокировано* определение новых слов через двоеточие.

По аналогии работы в диалоговом режиме с терминалом - *терминальном режиме*, при получении входного SMS (длина одного стандартного SMS не может превышать 160 символов), автоматически формируется выходной SMS с ответом форт-системы - это *дистанционный режим* форт-системы. Отличие этого режима от режима при работе с терминалом (*терминального режима*) заключается в том, что не работает слово WORDS. Кроме того, формирование выходного SMS с ответом форт-системы можно заблокировать. Данная возможность уменьшает затраты на эксплуатацию контролера и полезна для организации взаимодействия между контролерами - так называемого меж машинного взаимодействия (M2M). Для этого в тексте входного SMS сначала (или после пароля и по крайней мере одного пробела - если активирована защита паролем) нужно указать служебное слово NAK, отделенное от остального текста по крайней мере одним пробелом.

Кроме того, формирование выходного SMS с ответом форт-системы на входные SMS с *коротких номеров* (меньше 8 цифры) также заблокировано. Однако, это не мешает посылать управляющие SMS с бесплатных сайтов операторов мобильной связи и других автоматизированных серверов из сети интернет - просто такие SMS всегда будут без ответа.

Для организации взаимодействия с помощью SMS с автоматизированными системами мониторинга типа АСКУЭ, существует возможность идентифицировать каждое переданное и принятое SMS. Для этого в тексте входного SMS сначала (или после пароля и по крайней мере одного пробела - если активирована защита паролем) нужно указать служебное слово IDxxx, отделенное от остального текста по крайней мере одним пробелом. Параметр xxx это произвольное число в диапазоне от 0 до 65535, которое объединяется со служебным словом без пробела и является идентификатором сообщения. Форт-система пошлет в ответ SMS, в конце которого к стандартным словам (OK) или (ERROR - ...) будет добавлен данный идентификатор в виде (IDxxx OK) или (IDxxx ERROR - ...). Применение служебного слова IDxxx позволяет вести учет управляющих и принятых сообщений в условиях, когда операторы связи не гарантируют очередность передачи SMS через свою сеть.

Служебные слова IDxxx и NAK *не могут* быть применены одновременно - это во-первых не логично, а во-вторых приводит к ошибке интерпретации текста SMS.

Внимание: для данной версии интерпретатора языка ForthLogic™ существует ограничение относительно использования кириллических литер в текстах SMS - можно применять лишь латинские. Поэтому, при создании прикладного словаря для конкретной задачи, необходимо учитывать данный факт!

Для передачи текстовых сообщений в виде SMS, существует слово SMS которое работает следующим образом: из выходного буфера считывается номер телефона в международном или национальном формате и текст самого SMS, между которыми должен быть по крайней мере один пробел, дальше осуществляется попытка отправить SMS на указанный номер телефона, а на стек кладется логическое значение - результат выполнения слова SMS. При этом, логическое значение ИСТИНА означает, что попытка успешно завершилась, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM модуль выключен или занят). Приведем пример отправки SMS непосредственно из терминала:

```
 > ." +380507654321 Hello World! " SMS .
```

```
+380507654321 Hello World! -1 (OK)
>
```

Совокупная длина текста самого SMS не должна превышать 160 символов, в случае превышения этой длины система автоматически ее ограничит с отбрасыванием лишних символов. Текст длиной больше чем одна строки (77 символов) можно вывести в выходной буфер за несколько раз меньшими порциями. Приведем пример:

```
> : послатьSMS
." +380507654321 "
." Hello! Voltage AI2 = " 2 AI? F. ." AI3 = " 3 AI? F.
SMS DROP ;
(OK)
> 1 FPREC! послатьSMS
+380507654321 Hello! Voltage AI2 = 9.1 AI3 = 349.0 (OK)
>
```

Мы определили новое слово "послатьSMS", которое печатает в выходной буфер (и на терминал) номер телефона, пробел и определенным образом форматированный текст самого сообщения, после этого выполняется слово SMS, которое "потребляет" все эти данные и делает попытку послать SMS. Словом DROP мы отбрасываем результат работы слова SMS, поскольку его не анализируем.

Как видно из примера, в текст сообщения можно включать как строки-константы так и переменные данные - с помощью слов **.** (точка), **F.** (FLOAT - точка) и **FE.** (FLOAT ENGINEER - точка). При этом могут быть использованы все слова, которые кладут свои данные на стек данных или математический стек. Для форматирования текста сообщения можно использовать слова SPACE (пробел) и NEWLINE (новая строки).

Одна из форм коротких текстовых сообщений в сети сотовой связи GSM имеет название USSD (Unstructured supplementary service data) и используется операторами для выполнения разных сервисных заданий (подключение сервиса, пополнение счета, проверка баланса и тому подобное). Для работы с USSD запросами существует слово USSD которое работает следующим образом: из выходного буфера считывается номер телефона в международном или национальном формате и строки USSD запроса, между которыми должен быть по крайней мере один пробел, дальше осуществляется попытка отправить USSD запрос оператору сотовой связи, а на стек кладется логическое значение - результат выполнения слова USSD. При этом, логическое значение ИСТИНА означает, что попытка успешно завершилась, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM модуль выключен или осуществляется другой USSD запрос). Ответ оператора на удачный USSD запрос отправляется в виде SMS на указанный номер телефона. Совокупная длина USSD запроса не должна превышать 80 символов, в случае превышения этой длины система автоматически ее ограничит с отбрасыванием лишних символов.

Слово USSD можно использовать как дистанционно (в дистанционном режиме работы форт-системы в тексте SMS) так и локально (например, выполняя по таймеру для регулярной проверки баланса). Если при выполнении слова USSD осуществляется голосовой вызов, то действие слова будет задержано вплоть до окончания вызова. Приведем пример дистанционного использования USSD запроса для пополнения счета и проверки баланса:

1. Сформировать и послать первое SMS следующего содержания (указав при необходимости пароль и верную последовательность цифр из ваучера пополнения):

```
PASWORD NAK LAST ." *111*12345678909876# " USSD DROP
```

в ответ придет SMS с ответом оператора на USSD запрос (формат представления зависит от оператора и может быть пустым).

2. Сформировать и послать второе SMS следующего содержания (указав при необходимости пароль и верную последовательность цифр для проверки баланса):

```
PASWORD NAK LAST ." *123# " USSD DROP
```

в ответ придет SMS с ответом оператора на USSD запрос, в котором будет присутствовать информация о текущем балансе (формат представления зависит от оператора). В этом примере, для установки номера телефона, на который будет приходить ответ на USSD запрос, мы использовали слово LAST, тем самым указав номер телефона с которого высылались SMS. Телефон также можно указать непосредственно или с помощью слова USER.

Из технологических соображений, в *дистанционном режиме* работы форт-системы заблокировано непосредственное выполнение слов SMS и DIAL (о данном слове более детально в последующих разделах). Однако, это не касается слова USSD, которое спроектировано соответствующим образом. Слова SMS и DIAL могут быть выполнены в составе других слов как отложенные во времени задания.

Входные SMS сообщения и встроенный текстовый интерпретатор форт-системы естественным образом могут быть использованы для организации меж машинного взаимодействия между контролерами или между контролером и другими интеллектуальными устройствами, которые могут обрабатывать SMS. Для меж машинного взаимодействия между контролерами, одному контролеру необходимо сформировать и послать другому SMS, в котором указать необходимые действия или слова. При этом, в тексте такого SMS, сначала (или после пароля) необходимо указать служебное слово NAK - таким образом, другой контролер при обработке в текстовом интерпретаторе управляющих слов не сформирует и не отошлет автоматический ответ своей форт-системы, который может быть абсолютно "непонятен" форт-системе контролера, который инициировал обмен. Если в тексте выходного SMS не указать служебную команду NAK, другой контролер сформирует и отошлет автоматический ответ своей форт-системы, однако, такое SMS вообще не будет воспринято форт-системой контролера, который инициировал обмен - это мера предосторожности чтобы не создавалась ситуация с цепочкой бесконечных SMS между двумя контролерами.

Голосовые звонки

Для работы с голосовым каналом GSM модуля, реализована группа слов, которые позволяют осуществить и принять голосовой звонок, подключить внешнюю гарнитуру и настроить реакцию форт-системы на DTMF сигналы.

Воспроизведение сообщений

Для воспроизведения звукового сообщения в любой момент времени существования голосового звонка, существует слово PLAY (воспроизвести) которое работает следующим образом: из выходного буфера считывается строки - название файла в формате *.wav, который расположен на карте памяти SD/MMC (процедура создания звуковых файлов описана в приложении), из входного текста выбирается очередное слово обратного вызова (введенное после слова PLAY) и запускается процесс воспроизведения звукового сообщения, а на стек кладется логическое значение - результат выполнения слова PLAY. При этом, логическое значение ИСТИНА означает, что процесс

воспроизведения успешно начался, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, файл или карта памяти отсутствуют, уже осуществляется воспроизведение и тому подобное). Когда звуковое сообщение будет полностью воспроизведено, форт-система выполнит указанное после слова PLAY слово обратного вызова.

Для воспроизведения словами значения числа с вершины математического стека существует слово SAY (сказать) которое работает следующим образом: из выходного буфера считывается строки - путь к набору файлов чисел в формате *.wav, который расположен на карте памяти SD/MMC, с математического стека снимается верхнее значение, которое должно находиться в пределах от -999,9 до +999,9 включительно, из входного текста выбирается очередное слово обратного вызова (введенное после слова SAY) и запускается процесс воспроизведения словами значения числа, а на стек данных кладется логическое значение - результат выполнения слова SAY. При этом, логическое значение ИСТИНА означает, что процесс воспроизведения успешно начался, а НЕ ИСТИНА - что возникли аппаратные проблемы или число находится вне установленных пределов. Когда значение числа будет полностью воспроизведено, форт-система выполнит указанное после слова SAY слово обратного вызова. Путь к набору файлов для контролеров с дисплеем должен быть пустым (данные контролеры имеют встроенный набор файлов). Набор файлов чисел в формате *.wav описан в приложении.

По умолчанию, значения числа с вершины математического стека воспроизводится с указыванием знака числа и с точностью 1 цифра после десятичной запятой. Воспроизведение знака числа можно отключить с помощью слова NOAUTOSAYPLUS и повторно включить с помощью слова AUTOSAYPLUS.

Для прекращения воспроизведения звукового сообщения или числа с вершины математического стека существует слово MUTE, которое немедленно прекращает воспроизведение звука и блокирует выполнение заданного после слов PLAY или SAY слова обратного вызова.

При создании программы с использованием голосовых сообщений, для их более качественного воспроизведения, необходимо придерживаться определенных принципов, а именно: разбивать задачу на более мелкие фрагменты, которые должны выполняться с небольшими часовыми промежутками между ними. Также, с этой целью, форт-система *автоматически блокирует* выведение информации на дисплей и реакцию на клавиатуру во время воспроизведения любого голосового фрагмента.

Управление вызовом

Для определения статуса голосового вызова существует слово HOOK? (рычаг телефонного аппарата), которое кладет на стек логическое значение, которое отвечает состоянию голосового вызова. При этом, логическое значение ИСТИНА означает, что "трубка лежит на аппарате" - то есть, на данный момент не осуществляется ни один голосовой вызов, а НЕ ИСТИНА - что "трубка снята" - то есть, происходит голосовой вызов (входной или исходящий).

Для прекращения голосового вызова, существует слово HOLD (повесить тркбку) которое прекращает любой голосовой вызов (входной или исходящий).

Осуществление вызова

Для осуществления голосового вызова, существует слово DIAL (набрать номер) которое работает следующим образом: из выходного буфера считывается строки - номер телефона в международном или национальном формате, из входного текста выбирается два очередных слова обратного вызова (введенных после слова DIAL) и запускается процесс набора номера телефона, а на стек кладется логическое значение - результат выполнения слова DIAL. При этом, логическое значение ИСТИНА означает, что процесс дозвона успешно начался, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM модуль выключен или уже осуществляется вызов). Когда на втором конце телефонной линии будет снята трубка, форт-система, выполнит первое из указанных слов обратного

вызова, а в случае, когда никто не ответит в течение 60 секунд или вызов будет прерван абонентом, форт-система выполнит второе из указанных слов обратного вызова.

Теперь мы можем привести пример осуществления голосового вызова:

```
> : сообщить 1 AI? 2.345 F* SAY HOLD DROP ;
(OK)
> : поздороваться ." hello.wav " PLAY сообщить DROP ;
(OK)
> : позвонить ." +380501234567 " DIAL поздороваться STOP DROP ;
(OK)
> позвонить
+380501234567 (OK)
>
```

Сначала мы определили новое слово "сообщить", которое выполняет следующие действия: на вершину математического стека кладется значение измеренное на первом аналоговом входе, которое множится на масштабный коэффициент - таким образом на вершине математического стека будет находиться значение, например, температуры, которое нужно для выполнения слова SAY. Дальше выполняется слово SAY - начинается воспроизведение словами значения числа, в конце которого будет выполнено слово HOLD - положить трубку, а сразу после слова SAY будет выполнено слово DROP - чтоб отбросить со стека результат выполнения слова SAY, поскольку мы его не анализируем. Дальше, мы определили новое слово "поздороваться", которое выполняет следующие действия: в выходной буфер печатается название файла, которое необходимо для выполнения слова PLAY, дальше выполняется слово PLAY - начинается воспроизведение в конце которого будет выполнено слово "сообщить", а сразу после слова PLAY будет выполнено слово DROP - чтоб отбросить со стека результат выполнения слова PLAY, поскольку мы его не анализируем. Дальше мы определили новое слово "позвонить", которое выполняет следующие действия: в выходной буфер печатается номер телефона, который нужен для выполнения слова DIAL, дальше выполняется слово DIAL - начинается процесс дозвона, в конце которого будет выполнено или слово "поздороваться" - в случае, если трубку снимут, или слово STOP - в случае, если трубку не снимут в течении 60 секунд, а сразу после слова DIAL будет выполнено слово DROP - чтоб отбросить со стека результат выполнения слова DIAL, поскольку мы его не анализируем. Дальше мы просто выполняем слово "позвонить".

Прием вызова

Для приема голосового вызова, существует слово ANSWER (ответить) которое работает следующим образом: из выходного буфера считывается одна или несколько строк разделенных пробелом - номера телефонов в международном или национальном формате, из входного текста выбирается очередное слово обратного вызова (введенное после слова ANSWER) и осуществляется попытка ответить на вызов с указанных номеров телефонов, а на стек кладется логическое значение - результат выполнения слова ANSWER. При этом, логическое значение ИСТИНА означает, что попытка успешно завершилась и установлено голосовое соединение, а НЕ ИСТИНА - что на момент выполнения слова ANSWER не было входного вызова, голосовой вызов уже осуществляется или входной вызов выполнен с телефона, которого не было в списке. Когда попытка ответа успешно завершилась, форт-система выполнит указанное после слова ANSWER слово обратного вызова. Допустимо также не указывать никаких номеров телефонов, тогда ответ будет произведен на любой входной вызов который существует на момент выполнения слова ANSWER. Приведем пример:

```
> : поздороваться ." hello.wav " PLAY HOLD DROP ;
(OK)
> : ожидать ." +380501234567 +380507654321
ANSWER поздороваться
NOT IF 3.0 3 TIMER! ожидать THEN ;
(OK)
> ожидать
+380501234567 +380507654321 (OK)
>
```

Слово "поздороваться" мы уже встречали. Дальше мы определили новое слово "ожидать", которое выполняет следующие действия: в выходной буфер печатаются номера телефонов через пробел, которые нужны для выполнения слова ANSWER, дальше выполняется слово ANSWER - делается попытка ответить на возможный входной вызов, в конце которой, в случае наличия вызова с указанных телефонов, будет выполнено слово "поздороваться", а сразу после слова ANSWER будет осуществлен анализ выполнения слова ANSWER - если результат выполнения будет НЕ ИСТИНА, то через 3 секунды опять будет выполнено слово "ожидать", а если ИСТИНА, то ничего не будет выполнено. Дальше мы просто выполняем слово "ожидать", запуская процесс ожидания входного вызова вплоть до момента его успешного приема.

Иногда возникает необходимость выполнить какие-то действия в ответ лишь на факт существования входного голосового вызова без соединения. Такой способ управления обычно очень ограничен, но является бесплатным. Для приема голосового вызова без соединения, существует слово CLIP (аббревиатура, которая означает идентификацию по входному вызову), которое работает следующим образом: из выходного буфера считывается одна или несколько строк разделенных пробелом - номера телефонов в международном или национальном формате, из входного текста выбирается очередное слово обратного вызова (введенное после слова CLIP) и осуществляется анализ возможного входного вызова на предмет соответствия списку номеров телефонов, а на стек кладется логическое значение - результат выполнения слова CLIP. При этом, логическое значение ИСТИНА означает, что анализ успешен и голосовое соединение *разорвано*, а НЕ ИСТИНА - что на момент выполнения слова CLIP не было входного вызова, голосовой вызов уже осуществляется или входной вызов выполнен с телефона, которого не было в списке. Когда анализ успешно завершился, форт-система выполнит указанное после слова CLIP слово обратного вызова. Допустимо также не указывать никаких номеров телефонов, тогда данное слово будет выполнено в ответ на любой входной вызов, который существует на момент выполнения слова CLIP. Приведем пример:

```
> : перкл.бойлер 2 RO? NOT 2 RO! ;
(OK)
> : ожидать ." +380501234567 +380507654321
CLIP перкл.бойлер
NOT IF 3.0 3 TIMER! ожидать THEN ;
(OK)
> ожидать
+380501234567 +380507654321 (OK)
>
```

Список телефонов, на которые происходит попытка ответить словом ANSWER или CLIP, ограничивается длиной выходного буфера. Однако существует другой способ, который позволяет

существенно увеличить количество номеров телефонов с которых ожидаются входные вызовы. Приведем пример со словом ANSWER:

```
> : hello ." hello.wav " PLAY HOLD DROP ;
(OK)
> : wait1 ." 80501234567 " ANSWER hello DROP 3.0 1 TIMER! wait1 ;
(OK)
> : wait2 ." +380687654321 " ANSWER hello DROP 3.0 2 TIMER! wait2 ;
(OK)
> wait1 wait2
80501234567 +380687654321 (OK)
>
```

Слово hello аналогично слову "поздороваться" из предыдущего примера. Дальше мы определили два новых слов wait1 и wait2, которые выполняют подобные действия: в выходной буфер печатается номер телефона (свой для каждого слова), дальше выполняется слово ANSWER - делается попытка ответить на возможный входной вызов, в конце которого, в случае наличия вызова с указанного телефона, будет выполнено слово hello, а сразу после слова ANSWER будет выполнено слово DROP - чтобы отбросить со стека результат выполнения слова ANSWER. Через 3 секунды слова wait1 и wait2 выполнятся опять. Дальше мы просто вызываем слова wait1 и wait2, запуская бесконечный процесс ожидания входного вызова (это происходит даже во время осуществления голосового соединения - принцип роботы слова ANSWER предусматривает и такую возможность).

DTMF сигналы

При создании голосовых меню, необходимым элементом таких меню является программирование действий на нажатие клавиши мобильного телефона - то есть установка реакции на DTMF сигналы. Для этого существует три базовых слова ожидания: WAITKEY (ожидать клавишу), которое настраивает необходимую реакцию форт-системы на одиночные DTMF сигналы, WAITPW (ожидать пароль), которое настраивает необходимую реакцию форт-системы на введение с помощью DTMF сигналов обще системного пароля доступа и WAITSTR (ожидать строку), которое настраивает необходимую реакцию форт-системы на введение с помощью DTMF сигналов строки цифр, причем строки цифр может представлять как целое число так и число с десятичной запятой.

Для управления голосовым меню с мобильного телефона, пользователю доступны клавиши цифр от "1" до "9" а также клавиши "0", "*" и "#". В форт-системе им отвечают коды от 1 до 12 (то есть, клавишам от "1" до "9" отвечает код от 1 до 9, клавише "0" - 10, клавише "*" - 11, а клавише "#" - 12). По умолчанию, каждое нажатие клавиши сопровождается соответствующим сигналом в динамике мобильного телефона - это значит, что форт-система восприняла факт нажатия. Такое поведение системы можно отключить с помощью слова NODTMFCONFIRM (выключить подтверждение DTMF сигналов) или повторно включить с помощью слова DTMFCONFIRM (включить подтверждение DTMF сигналов), причем данная опция не сохраняется при выключении питания. Необходимость в отключении подтверждения DTMF сигналов может возникнуть в случае проблем с передачей сигналов у некоторых операторов связи и в случае повышенной секретности при наборе команд или кодов.

Также можно в произвольный момент времени осуществления голосового соединения генерировать в динамике мобильного телефона произвольный тон соответствующей клавиши с помощью слова TONE. Слово TONE работает следующим образом: со стека снимается верхнее значение - код клавиши в диапазоне от 1 до 12 и в динамике мобильного телефона происходит генерация одиночного DTMF сигнала с тоном, который отвечает коду клавиши.

Рассмотрим работу базовых слов ожидания. Слово WAITKEY (ожидать клавишу) работает следующим образом: из входного текста выбирается три очередных слова обратного вызова (введенных после слова WAITKEY) и настраивается реакция форт-системы на DTMF сигналы, а на стек кладется логическое значение - результат выполнения слова WAITKEY. При этом, логическое значение ИСТИНА означает, что процесс настройки произведен успешно, а НЕ ИСТИНА - что возникли проблемы (например, на данный момент не осуществляется ни один голосовой вызов). Когда на втором конце телефонной линии будет нажата одна из клавиш мобильного телефона, форт-система выполнит первое из указанных слов обратного вызова, причем, на момент выполнения данного слова, на вершине стека данных будет находиться код последнего полученного DTMF сигнала - то есть код нажатой клавиши; в случае, когда ничего не будет нажато на протяжении 60 секунд, форт-система выполнит второе из указанных слов обратного вызова; и, наконец, когда связь будет прервана пользователем, форт-система выполнит третье из указанных слов обратного вызова. Приведем пример простого голосового меню, которое состоит из одного пункта:

```
> : анализ_кнопки 1 = IF ." true.wav " PLAY HOLD
ELSE ." by.wav " PLAY HOLD THEN DROP ;
(OK)
> : ждать_кнопку WAITKEY анализ_кнопки HOLD STOP DROP 2 TONE ;
(OK)
> : поздороваться ." hello.wav " PLAY ждать_кнопку DROP ;
(OK)
> : голосовое_меню ." +380501234567 " DIAL поздороваться STOP DROP ;
(OK)
> голосовое_меню
+380501234567 (OK)
>
```

Мы определили новое слово "анализ_кнопки", которое выполняет следующие действия: сравнивает код последнего полученного DTMF сигнала с 1, если в результате сравнения на стеке будет логическое значение ИСТИНА, то воспроизводится сообщение из файла "true.wav" и связь обрывается, иначе воспроизводится сообщение из файла "by.wav" и связь также обрывается, в любом случае, будет также выполнено слово DROP - чтоб отбросить со стека результат выполнения любого из слов PLAY, поскольку мы его не анализируем. Дальше мы определили новое слово "ждать_кнопку", которое настраивает необходимую реакцию форт-системы на DTMF сигналы и генерирует DTMF сигнал приглашения: при нажатии любой кнопки, будет выполнено слово "анализ_кнопки", в случае, когда пройдет установленное время реакции 60 секунд, связь будет разорвана форт-системой, а в случае, когда связь будет разорвана пользователем голосового меню, не будет никакой реакции. Дальше мы определили новое слово "поздороваться", которое воспроизводит сообщение из файла "hello.wav", после чего выполняет слово ждать_кнопку. Дальше было определено главное слово - "голосовое_меню", которое осуществляет голосовой звонок на указанный телефон и, в случае успеха, выполняет слово "поздороваться". Выполнение главного слова в следующей строке начинает процесс воспроизведения голосового меню.

Слово WAITPW (ожидать пароль) работает подобным слову WAITKEY образом: из входного текста выбирается три очередных слова обратного вызова (введенных после слова WAITPW) и настраивается реакция форт-системы на введение обще системного пароля доступа с помощью DTMF сигналов, а на стек кладется логическое значение - результат выполнения слова WAITPW. При этом, логическое значение ИСТИНА означает, что процесс настройки произведен успешно, а НЕ

ИСТИНА - что возникли проблемы (например, на данный момент не осуществляется ни один голосовой вызов). Когда на втором конце телефонной линии будет введен верный пароль (при этом каждую следующую цифру следует вводить только после сигнала приглашения и завершается введение кнопкой "#"), форт-система выполнит первое из указанных слов обратного вызова; в случае, когда ничего не будет нажато на протяжении 60 секунд или введен не вереный пароль, форт-система выполнит второе из указанных слов обратного вызова; и, наконец, когда связь будет прервана пользователем, форт-система выполнит третье из указанных слов обратного вызова. Изменим предыдущий пример простого голосового меню, введя проверку пароля (нововведения помечены темно серым цветом):

```
> : анализ_кнопки 1 = IF ." true.wav " PLAY HOLD
ELSE ." by.wav " PLAY HOLD THEN DROP ;
(OK)
> : ждать_кнопку WAITKEY анализ_кнопки HOLD STOP DROP 2 TONE ;
(OK)
> : поздороваться ." hello.wav " PLAY ждать_кнопку DROP ;
(OK)
> : ждать_пароль WAITPW поздороваться HOLD STOP DROP 6 TONE ;
(OK)
> : пароль ." parol.wav " PLAY ждать_пароль DROP ;
(OK)
> : голосовое_меню
." +380501234567 " DIAL пароль STOP DROP ;
(OK)
> голосовое_меню
+380501234567 (OK)
>
```

Слово WAITSTR (ожидать строку) также работает подобным слову WAITKEY образом: из входного текста выбирается три очередных слова обратного вызова (введенных после слова WAITSTR) и настраивается реакция форт-системы на введение строки цифр с помощью DTMF сигналов, а на стек кладется логическое значение - результат выполнения слова WAITSTR. При этом, логическое значение ИСТИНА означает, что процесс настройки произведен успешно, а НЕ ИСТИНА - что возникли проблемы (например, на данный момент не осуществляется ни один голосовой вызов). Когда на втором конце телефонной линии будет введена строки цифр, форт-система выполнит первое из указанных слов обратного вызова, причем, на момент выполнения данного слова, введенное число будет находиться на вершине математического стека; в случае, когда ничего не будет нажато на протяжении 60 секунд, форт-система выполнит второе из указанных слов обратного вызова; и, наконец, когда связь будет прервана пользователем, форт-система выполнит третье из указанных слов обратного вызова. При введении строки цифр, каждую последующую цифру следует вводить только после сигнала успешного введения предыдущей цифры, для введения десятичной запятой используется кнопка "*", а завершать введение необходимо кнопкой "#". Приведем пример простого голосового меню введения значения температуры в математическую переменную:

```
> : уст_темп FDUP 60.0 F<
IF 1 FVAR! ." pidtvr.wav " PLAY HOLD DROP
ELSE ." nevirno.wav " PLAY HOLD DROP
```

```
THEN;
(OK)
> : ждать_темп WAITSTR уст_темп HOLD STOP DROP 11 TONE ;
(OK)
> : поздороваться ." zaprosh.wav " PLAY ждать_темп DROP ;
(OK)
> : голосовое_меню ." +380501234567 " DIAL поздороваться STOP DROP ;
(OK)
> голосовое_меню
+380501234567 (OK)
>
```

Мы определили новое слово "уст_темп", которое выполняет следующие действия: делает копию положенного на математический стек числа, сравнивает его значение с числом 60.0 (убирая при этом с математического стека одну из копий), если в результате сравнения на стеке данных будет положено логическое значение ИСТИНА, то происходит присвоение значения температуры математической переменной номер 1 (при этом с математического стека убирается последняя копия), воспроизводится сообщение из файла "pidtvr.wav" и связь обрывается, иначе воспроизводится сообщение из файла "nevirno.wav" и связь также обрывается. Дальше мы определили новое слово "ждать_темп", которое настраивает необходимую реакцию форт-системы на введение строки цифр: при нажатии кнопки "#", будет выполнено слово "уст_темп", в случае, когда пройдет установленное время реакции 60 секунд, связь будет разорвана форт-системой, а в случае, когда связь будет разорвана пользователем голосового меню, не будет никакой реакции. Дальше мы определили новое слово "поздороваться", которое воспроизводит сообщение из файла "zaprosh.wav", после чего выполняет слово "ждать_темп". Дальше было определено главное слово - "голосовое_меню", которое осуществляет голосовой звонок на указанный телефон и, в случае успеха, выполняет слово "поздороваться". Выполнение главного слова в следующей строке начинает процесс воспроизведения голосового меню.

Гарнитура

Для подключения внешней гарнитуры (комплекта оборудования для осуществления разговора) во время осуществления голосового вызова (входного или исходящего) существует слово MIC (гарнитура). Выполнение данного слова приводит к переключению аудио канала GSM модуля с цифрового синтезатора на внешнюю гарнитуру присоединенную или к входам SPK, MIC+, MIC-, или к аудио разъему MIC/EAR. При этом на стек кладется логическое значение - результат выполнения слова MIC: логическое значение ИСТИНА означает, что процесс переключения произведен успешно, а НЕ ИСТИНА - что возникли проблемы (например, на данный момент не осуществляется ни один голосовой вызов). При успешном выполнении слова MIC, воспроизведение голосовых сообщений, становится невозможным - аудио канал GSM модуля будет работать с внешним источником звука а во внешнем динамике будет воспроизводиться аудио из GSM модуля. После того, как связь будет прервана (пользователем или форт-системой), аудио канал GSM модуля автоматически переключается на цифровой синтезатор - то есть при следующем голосовом вызове опять становится доступным воспроизведение голосовых сообщений а аудио из GSM модуля не будет слышно во внешнем динамике. Также аудио канал GSM модуля можно переключить в любой момент времени с помощью слова VOICE. Выполнение данного слова приводит к переключению аудио канала GSM модуля на цифровой синтезатор. При этом на стек кладется логическое значение - результат выполнения слова VOICE: логическое значение ИСТИНА означает, что процесс переключения произведен успешно, а НЕ ИСТИНА - что возникли проблемы (например, на данный момент не

осуществляется ни один голосовой вызов). Прием DTMF сигналов возможен как при работе GSM модуля от цифрового синтезатора, так и при работе от внешней гарнитуры.

Для регулировки чувствительности внешней гарнитуры существует слово MICLEVEL (чувствительность гарнитуры) которое работает следующим образом: с вершины стека данных снимается число - новое *условное значение чувствительности* в диапазоне от 0 до 15 и устанавливается заданная чувствительность. Увеличение или уменьшение условной чувствительности на 1 приводит к увеличению или уменьшению реальной чувствительности гарнитуры в 1,1885 раз. По умолчанию, условная чувствительность равна 2 а ее новое значение не сохраняется при выключении питания.

Для регулировки громкости внешней гарнитуры существует слово SPKLEVEL (громкость гарнитуры) которое работает следующим образом: с вершины стека данных снимается число - новое *условное значение громкости* в диапазоне от 0 до 100 и устанавливается заданная громкость. Уменьшение условной громкости приводит к уменьшению реальной громкости в динамике и наоборот. По умолчанию, условная громкости ровна 100 а ее новое значение не сохраняется при выключении питания.

Передача данных CSD

Прием и передача данных с помощью канала CSD происходит по принципу модемного соединения точка-точка. На одном конце канала, при этом, находится контролер серии ES-ForthLogic™ со встроенным модулем GSM. На втором конце данного канала может быть либо обычный аналоговый модем, который работает в проводной телефонной сети, либо любой GSM модем, который поддерживает протокол CSD, либо другой контролер серии ES-ForthLogic™ со встроенным модулем GSM.

Аналогично входящим SMS, входной поток данных из канала CSD непосредственно попадает во входной буфер текстового интерпретатора форт-системы и, соответственно, интерпретируется и выполняется. Как уже упоминалось, данная концепция чрезвычайно гибкая, однако, в данном случае, она более мощная чем аналогичная концепция входящих SMS потому, что во входном потоке *разрешено* определение новых слов через двоеточие. Это сделано так потому, что пользователю системы получить доступ к каналу CSD с помощью обычного телефона без применения компьютера и специального программного обеспечения практически невозможно, и соответственно вероятность случайно нарушить роботу системы практически отсутствует. Однако для систем типа M2M - меж машинного взаимодействия или систем мониторинга типа АСКУЭ такая опция, как дистанционное определение новых слов через двоеточие, может пригодиться.

При получении входного потока данных с помощью канала CSD (длина одной порции данных не может превышать 160 символов), автоматически формируется выходной поток данных с ответом форт-системы - это *CSD режим* форт-системы. Отличие этого режима от режима при работе с терминалом (*терминального режима*) заключается в том, что не работает слово WORDS, а в выходном потоке отсутствуют все сообщения об ошибках и информация в квадратных скобках генерируемая некоторыми словами (".S", ".FS" и тому подобными), отсутствуют все текстовые сообщения форт-системы в круглых скобках ("OK", "ERROR-..." и тому подобные) а также отсутствует символ приглашения форт-системы (">"). Это позволяет в режиме on-line направлять ответ одной форт-системы непосредственно на вход другой, что упрощает построение системы типа M2M. Однако, в случае возникновения ошибки, сообщения о них отображаются в терминальном режиме при условии что существует активное подключение к терминалу.

Из программы на языке ForthLogic™ можно инициировать подключение к каналу CSD, отправить данные в виде строки текста, инициировать выполнение любого известного слова в случае получения данных через канал, разорвать соединение и посмотреть состояние соединения. Для этого реализован ряд слов которые представлены далее по тексту.

Для осуществления подключения к каналу CSD существует слово CONNECTCSD (подключиться к каналу CSD) которое работает следующим образом: из выходного буфера считывается строки - номер телефона в международном или национальном формате, из входного текста выбирается два очередных слова обратного вызова (введенных после слова CONNECTCSD) и запускается процесс набора номера телефона, а на стек кладется логическое значение - результат выполнения слова CONNECTCSD. При этом, логическое значение ИСТИНА означает, что процесс подключения успешно начался, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM модуль выключен или уже осуществляется другой исходящий или входящий вызов). Когда успешно образуется канал CSD, форт-система выполнит первое из указанных слов обратного вызова, в случае, когда канал не сможет образоваться на протяжении 60 секунд или не сможет образоваться в принципе (попытка образовать канал не с модемом), форт-система выполнит второе из указанных слов обратного вызова.

Для передачи сообщения в канал CSD существует слово SENDCSD (отправить данные в канал CSD) которое работает следующим образом: из выходного буфера считывается строки сообщения, из входного текста выбирается очередное слово обратного вызова (введенное после слова SENDCSD) и строки передается через канал CSD, а на стек кладется логическое значение - результат выполнения слова SENDCSD. При этом, логическое значение ИСТИНА означает, что процесс передачи успешно начался, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM модуль выключен или канал CSD закрыт). В случае, когда в ответ придет сообщение со второго конца канала CSD, оно попадет сначала в интерпретатор форта-системы (см. выше по тексту), после чего форт-система выполнит указанное после слова SENDCSD слово обратного вызова.

Для разрыва канала CSD в любой момент времени существует слово BREAKCSD (разорвать канал CSD), а для получения статуса канала существует слово STATUSCSD (состояние канала CSD), которое возвращает на стек логическое значение ИСТИНА если канал передачи данных CSD открыт и логическое значение НЕ ИСТИНА в противоположном случае.

Приведем пример использования вышеупомянутых слов для построения системы типа M2M - меж машинного взаимодействия, в котором образуется канал CSD между двумя контролерами. Дальше один из них (ГЛАВНЫЙ) посылает запросы и обрабатывает ответы таким образом, что состояние цифровых входов другого контролера (ПОДЧИНЕННОГО) отображается на цифровых выходах ГЛАВНОГО:

```
> 0 CONSTANT send_a
(OK)
> : (send) send_a EXECUTE ;
(OK)
> : read 4 DO! 3 DO! 2 DO! 1 DO! 0.1 1400 BEEP (send) ;
(OK)
> : send ." 5 DI? . 6 DI? . 7 DI? . 8 DI? . 0.1 2400 BEEP "
SENDCSD read  DROP ;
(OK)
> ." send " FIND TO send_a
(OK)
> : start 1 USER CONNECTCSD send STOP DROP ;
(OK)
> : stop BREAKCSD ;
(OK)
>
```

Данная программа записывается в память только одного контролера - ГЛАВНОГО. Дальше ГЛАВНЫЙ контролер инициирует соединение с ПОДЧИНЕННЫМ и начинает управляемый обмен состоянием цифровых входов. Рассмотрим данный пример детальнее.

Канал данных открывается словом "start", как только это происходит выполняется слово "send", которое посылает запрос "5 DI? . 6 DI? . 7 DI? . 8 DI? . 0.1 2400 BEEP". Согласно этому запросу, другой контролер - ПОДЧИНЕННЫЙ выводит в свой выходной буфер состояние цифровых входов в виде цифр 0 или -1 и воспроизводит короткий звуковой сигнал. Дальше все содержимое его выходного буфера попадает в канал CSD в качестве ответа. Этот ответ сначала попадает во входной буфер ГЛАВНОГО контролера, который интерпретирует цифры и кладет их на свой стек. После этого форт-системой выполняется слово "read", которое снимает цифры со стека и записывает их в соответствующие цифровые выходы. Дальше оно воспроизводит короткий звуковой сигнал и опять выполняет слово "send" (через механизм обратной ссылки, см. раздел "Векторное выполнение") - таким образом весь цикл запрос-ответ повторяется бесконечное количество раз.

Канал CSD также можно использовать для неконтролируемого из программы на языке ForthLogic™ обмена данными с устройствами, присоединенными к последовательному интерфейсу RS485. Фактически образуется "прозрачный" *модемный канал* между устройством и программой для ПК, которая инициировала данный канал с помощью обычного модема или GSM-модема, и не накладываются никакие ограничения на протокол обмена по данному каналу. Однако, следует принять во внимание временную задержку, которая возникает из-за сути коммутированных по времени каналов передачи данных в сетях GSM. Практически, это заключается в том, что в непрерывном потоке данных возможны паузы до 20 мс. Также следует помнить, что скорость обмена фиксирована и составляет 9600 бит/сек.

Для образования модемного канала применяется слово MODEMCSD. Данное слово можно применять лишь тогда, когда предварительно образован канал передачи данных CSD: оно возвращает на стек логическое значение ИСТИНА если модемный канал успешно образовался и логическое значение НЕ ИСТИНА в противоположном случае. Практически, данное слово можно применить как в терминальном режиме (из программы на ForthLogic™ или непосредственно с терминала) так и в CSD режиме форт-системы (например, из программы для ПК, которая инициирует модемный канал).

Когда образуется модемный канал, все другие способы обмена через GSM (голосовые сообщения, SMS, GPRS) не работают. Канал можно разорвать путем разрыва самого CSD соединения с противоположного конца линии. При этом возобновляется работа через остальные каналы GSM (голосовые сообщения, SMS, GPRS и CSD в обычном режиме).

Передача данных GPRS

GPRS режим форт-системы

Прием и передача данных с помощью канала GPRS происходит по принципу клиент-сервер с использованием протокола TCP. При обычном клиент-серверном соединении, над протоколом TCP выполняется текстовый протокол обмена с форт-системой подобный обычному терминальному режиму работы: при получении входного потока данных (длина одной порции данных не может превышать 160 символов), автоматически формируется исходящий поток данных с ответом форт-системы - это *GPRS режим* форт-системы. При этом, в исходящем потоке отображаются все ошибки и служебная информация форт-системы, что не приемлемо в случае меж машинного взаимодействия (M2M) между двумя контролерами. Для отключения отображения служебной информации и ошибок существует слово M2MTCPON (включить режим M2M через протокол TCP). Повторно включить нормальный режим позволяет слово M2MTCPOFF (выключить режим M2M через протокол TCP). При пропадании питания данная опция не сохраняется, а при подаче питания по умолчанию устанавливается нормальный режим передачи данных через протокол TCP.

Принципы клиент-серверного соединения

Необходимо детально остановиться на технологии клиент-сервер. На одном конце канала всегда находится *сервер* а на втором *клиент* и, поэтому, существует два сценария соединения. Первый сценарий заключается в том, что в качестве сервера удобно использовать серверную программу которая работает на ПК, который имеет фиксированный IP-адрес во всемирной сети INTERNET. В этом случае контролер серии ES-ForthLogic™ со встроенным модулем GSM выступает в качестве клиента, который инициирует соединение с сервером. Преимущество такого сценария состоит в том, что к серверу одновременно может быть подсоединено очень много клиентов (сотни и тысячи) и нет необходимости в применении специальных SIM-карт с доступными извне IP-адресами.

Второй сценарий заключается в том, что в качестве сервера выступает контролер серии ES-ForthLogic™ со встроенным модулем GSM, который может одновременно принять лишь *одно* клиентское соединение. При этом необходимо гарантировать доступность IP-адреса контролера во всемирной сети INTERNET - это достигается при применении специальных SIM-карт и специальных телеметрических тарифов оператора сети GSM. Преимущество данного сценария заключается в том, что на втором конце можно использовать клиентскую программу которая работает на ПК и имеет любой доступ к сети INTERNET. Такая программа должна одновременно образовывать столько клиентских соединений с контроллерами, сколько необходимо.

Слова для работы с GPRS

Из программы на языке ForthLogic™ можно инициировать подключение к GPRS, установить соединение в режиме клиента или сервера, отправить данные в виде строки текста, разорвать соединение и посмотреть состояние соединения. Для этого реализован ряд слов, которые представлены далее по тексту. Также реализована поддержка передача данных, которые накапливаются во время работы регистратора, через существующее GPRS-соединение либо по протоколу TCP/IP, либо на сервер баз данных MYSQL с применением протокола HTTP - более детально это описано в соответствующем разделе. Кроме работы в режиме клиента или сервера с другим контролером или программой которая работает на ПК, существует возможность поддерживать постоянное соединение со специализированным OPC-сервером. При этом применяется первый сценарий технологии клиент-сервер (описанный выше).

Начальные настройки

Для работы с GPRS, прежде всего необходимо наличие данной услуги у оператора сети GSM на используемой SIM-карте. Для выяснения возможности подключения к сервису GPRS, существует слово STATUSGPRS (состояние сервиса GPRS), которое возвращает на стек логическое значение ИСТИНА, если сервис GPRS для данной SIM-карты активирован. Если сервис GPRS не активируется по умолчанию (свойство тарифа) или случился сбой сети, то для активации сервиса существует слово ATTACHGPRS (подключить сервис GPRS), которое делает попытку подключиться к сервису GPRS а на стек возвращает логическое значение ИСТИНА, если попытка началась удачно и НЕ ИСТИНА если возникли проблемы (например, GSM модуль выключен). Процесс подключения сервиса может занимать до 10 секунд и проверяется с помощью слова STATUSGPRS.

Для хранения параметров канала GPRS таких как APN и IP в энергонезависимой памяти, существуют специальные одноименные строчные переменные. Длина переменных IP и APN составляет 70 символов. Для работы с этими переменными существуют слова IP, APN, SETIP и SETAPN. Слова IP и APN печатают в выходном буфере и на терминале содержимое одноименной переменной. А слова SETIP и SETAPN запоминают текст из выходного буфера в соответствующей переменной IP и APN. В зависимости от режима работы канала GPRS, семантика содержимого этой переменной, может изменяться.

Для хранения параметров работы со специализированным OPC-сервером или сервером баз данных MYSQL в энергонезависимой памяти, также существуют специальные строчные переменные ID (идентификатор) и URL (адрес). Переменная ID предназначена для хранения имени и пароля для доступа к отдаленному серверу. Имя и пароль должны быть разделены пробелом и иметь длину не более чем 15 символов. Переменная URL предназначена для указания адреса внутри файловой системы сервера, к которому происходит обращение с применением протокола HTTP. Длина переменной ID составляет 30 символов, а переменной URL - 70 символов. Для работы с этими переменными существуют слова ID, URL, SETID и SETURL. Слова ID и URL печатают в выходном буфере и на терминале содержимое одноименной переменной. А слова SETID и SETURL запоминают текст из выходного буфера в соответствующей переменной ID и URL.

Для конфигурации протокола работы TCP и подключения по GPRS, существует слово CONFIGTCP (конфигурация протокола TCP), которое используется вместе с системными константами CLIENT (клиент), CLIENTDB (клиент сервера баз данных MYSQL), CLIENTSQL (поточный клиент сервера баз данных MYSQL), CLIENTOPC (клиент OPC-сервера) и SERVER (сервер) и работает следующим образом: со стека снимается число, которое отвечает одной из вышеупомянутых системных констант, из выходного буфера считывается строки которая состоит из APN, имени и пароля разделенных пробелами и происходит настройка протокола работы TCP с подключением к GPRS. На стек при этом кладется логическое значение - результат выполнения слова CONFIGTCP: логическое значение ИСТИНА означает, что процесс настройки и подключения осуществился успешно, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM модуль выключен или отсутствует сервис GPRS). APN который указывается в выходном буфере, предоставляется оператором для вхождения в сеть GPRS. Иногда также предоставляется имя и пароль, однако, если они отсутствуют, то строки состоит только из APN.

Для установки номера порта, который будет "прослушиваться" в режиме работы типа SERVER, существует слово CONFIGPORT, которое работает следующим образом: со стека снимается число в диапазоне от 1 до 65535 и происходит установка номера порта. На стек при этом кладется логическое значение - результат выполнения слова CONFIGPORT: логическое значение ИСТИНА означает, что процесс установки осуществился успешно, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM модуль выключен или отсутствует сервис GPRS). Номер порта по умолчанию всегда равняется 2020 и его установка имеет смысл только для работы в режиме SERVER. При пропадании питания номер порта не сохраняется.

К ресурсам в сети INTERNET можно обращаться через их IP-адреса (например 124.34.1.145) или через предназначенные для них URL-адреса (например: www.es.ua). По умолчанию, установлен режим IP-адресов - так удобнее для телеметрических систем. Однако, в такой задаче как работа с сервером баз данных MYSQL, возникает необходимость указывать URL-адреса. Для установки режима адресации с помощью URL-адресов существует слово DNSMODE (режим имен), которое работает следующим образом: из выходного буфера считывается строки, которая состоит из двух IP-адресов DNS-серверов разделенных пробелами и происходит установка режима URL-адресов. На стек при этом возвращается результат выполнения слова DNSMODE: логическое значение ИСТИНА означает, что процесс установки осуществился успешно, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM модуль выключен или отсутствует сервис GPRS). Допускается не указывать IP-адреса DNS-серверов, тогда для поиска URL-адресов система обращается к стандартным открытым DNS-серверам 208.67.220.220 и 208.67.222.222. Для установки режима адресации с помощью IP-адресов предназначено слово IPMODE (режим IP-адресов), которое возвращает на стек логическое значение - результат своего выполнения: логическое значение ИСТИНА означает, что процесс установки осуществился успешно, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM модуль выключен или отсутствует сервис GPRS). При пропадании питания опция режима адресации не сохраняется.

Настройка и разрыв соединения

Для установки соединения согласно протоколу TCP, существует слово OPENTCP (установить соединение по протоколу TCP), которое работает следующим образом: если протокол был настроен на работу в режиме CLIENT или CLIENTOPC, то из выходного буфера считывается строки которая состоит из IP-адреса и номера порта разделенных пробелом, если протокол был настроен на работу в режиме CLIENTSQL, то из выходного буфера ничего не считывается, а параметры IP-адреса и номера порта хранятся в переменной IP в энергонезависимой памяти, из входного текста выбирается два очередных слова обратного вызова (введенных после слова OPENTCP) и происходит настройка соединения. На стек при этом кладется логическое значение - результат выполнения слова OPENTCP: логическое значение ИСТИНА означает, что процесс соединения начался успешно, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM модуль выключен или отсутствует сервис GPRS). Когда успешно установится соединение согласно протоколу TCP, форт-система выполнит первое из указанных после слова OPENTCP слов обратного вызова, в случае, когда соединение не сможет образоваться в течение 120 секунд, форт-система выполнит второе из указанных слов обратного вызова. Адреса в выходном буфере указываются только для режимов соединения типа CLIENT или CLIENTOPC и являются адресами сервера с которым происходит попытка соединения. Для режима соединения типа SERVER данное слово лишь запускает сервер, который ожидает соединения со стороны клиента.

Для передачи сообщения согласно протоколу TCP в любом обычном режиме соединения типа CLIENT или SERVER, существует слово SENDTCP (отправить данные по протоколу TCP) которое работает следующим образом: из выходного буфера считывается строки сообщения и передается по протоколу TCP, а на стек кладется логическое значение - результат выполнения слова SENDTCP. При этом, логическое значение ИСТИНА означает, что процесс передачи успешно осуществился, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM модуль выключен, отсутствует сервис GPRS или канал GPRS закрыт). Максимальная длина сообщения составляет 160 символов.

Передача сообщений в режиме соединения типа CLIENTOPC является *невозможной*, поскольку обмен данными с OPC-сервером происходит по специализированному внутреннему протоколу. OPC-сервер делает запросы а система автоматически и прозрачно для пользователя осуществляет обработку и формирование ответа. Передача сообщений в режимах соединения типа CLIENTDB или CLIENTSQL также является невозможной, поскольку данные передаются автоматически из внутренней энергонезависимой или оперативной памяти на сервер баз данных MYSQL с применением протокола HTTP. Данный процесс начинается словами LOG>DB или LOG>SQL и полностью контролируется системой.

Для разрыва соединения типа CLIENT, CLIENTSQL или CLIENTOPC в любой момент времени, существует слово CLOSETCP (разорвать соединение согласно протоколу TCP). После разрыва соединения типа CLIENT, CLIENTSQL или CLIENTOPC, можно осуществить попытку соединения с другим сервером.

Для отключения от GPRS в любой момент времени, существует слово SHUTTCP (разорвать подключение). После отключения от GPRS, необходимо опять подключаться с помощью слова CONFIGTCP.

Статус соединения

Для получения статуса соединения согласно протоколу TCP, существует слово STATUSTCP (состояние соединения согласно протоколу TCP), которое возвращает на стек логическое значение ИСТИНА если в режиме соединения типа CLIENT, CLIENTDB, CLIENTSQL или CLIENTOPC, существует физическое соединение с отдаленным сервером.

В режиме соединения типа SERVER слово STATUSTCP возвращает на стек логическое значение ИСТИНА если сам сервер является активным (при этом может отсутствовать само соединение с

клиентом). Для получения статуса соединения с клиентом в режиме соединения типа SERVER, существует слово STATUSSERVER (состояние сервера), которое возвращает на стек логическое значение ИСТИНА если в режиме соединения типа SERVER существует физическое соединение с отдаленным клиентом.

Для получения статуса работы с OPC-сервером в режиме соединения типа CLIENTOPC, существует слово STATUSOPC (состояние OPC), которое возвращает на стек логическое значение ИСТИНА если в режиме соединения типа CLIENTOPC существует логическое соединение с OPC-сервером. Логическое соединение с OPC-сервером означает постоянный автоматический обмен данными по специализированному внутреннему протоколу.

После подключения к GPRS, можно узнать собственный IP-адрес, который автоматически предоставляет оператор, с помощью слова LOCALIP (собственный IP-адрес). Данное слово печатает в выходной буфер и на терминал строку собственного IP-адреса.

Если в режиме соединения типа SERVER существует физическое соединение с удаленным клиентом, то можно узнать IP-адрес клиента с помощью слова REMOTEIP (IP-адрес клиента). Данное слово печатает в выходной буфер и на терминал строку IP-адреса клиента. Также в режиме соединения типа SERVER можно ограничить круг удаленных клиентов, если записать в энергонезависимую переменную IP IP-адрес клиента которому разрешено подключаться к серверу. Если в эту переменную записать пустую строку, то к серверу могут подключаться все клиенты без ограничений.

Особенность режима CLIENTOPC

Как уже упоминалось, в режиме CLIENTOPC обмен данными с OPC-сервером происходит автоматически по специализированному внутреннему протоколу. OPC-сервер это специализированный сервер который, с одной стороны, осуществляет обмен данными с контролером через канал GPRS а, с другой стороны, предоставляет доступ к этим данным OPC-клиентам, в качестве которых чаще всего выступают разнообразные системы дистанционного управления и мониторинга технологических процессов (АСУТП, SCADA). OPC-сервер поддерживает спецификации OPC DA v1 и v2 и должен работать на ПК, который имеет фиксированный IP-адрес во всемирной сети INTERNET. Во время работы с OPC-сервером контролер выступает в качестве клиента и поэтому можно применять обычные SIM-карты с поддержкой GPRS-соединения. Также можно применять SIM-карты с поддержкой VPN. Инициатором соединения с OPC-сервером выступает контролер. Для успешного соединения, контролер должен пройти аутентификацию, которая заключается в совпадении имени и пароля, которые задаются с помощью переменной ID. Одновременно к одному OPC-серверу может быть подсоединено множество контролеров, что позволяет создавать распределенную систему дистанционного управления и мониторинга технологических процессов.

Принцип обмена данными с OPC-сервером заключается в том, что OPC-сервер делает запросы а система автоматически и прозрачно для пользователя осуществляет обработку запросов и формирование ответов. При этом OPC-сервер может свободно устанавливать и читать все глобальные переменные, глобальные переменные в формате чисел с плавающей запятой и глобальные битовые переменные. Программа на языке ForthLogic™ не имеет непосредственного доступа к обмену данными с OPC-сервером, однако может взаимодействовать с OPC-сервером (а через него и с системой АСУТП или SCADA) через глобальные переменные.

Особенность режимов CLIENTDB и CLIENTSQL

В режиме соединения типа CLIENTDB или CLIENTSQL данные передаются автоматически из внутренней энергонезависимой или оперативной памяти на сервер баз данных MYSQL с применением протокола HTTP. Как уже упоминалось, данный процесс начинается словами LOG>DB или LOG>SQL и полностью контролируется системой.

Принцип передачи данных на сервер баз данных MYSQL заключается в том, что система сначала передает файл из внутренней энергонезависимой или оперативной памяти на HTTP-сервер с помощью метода POST. Этот файл принимается и обрабатывается специализированным скриптом, имя которого необходимо указывать в переменной URL. Дальше этот файл анализируется и данные из него передаются в таблицу на сервер баз данных MYSQL. Отличие режимов соединений типа CLIENTDB от CLIENTSQL заключается в том, что в режиме типа CLIENTDB HTTP-сервер при получении файла закрывает физическое соединение с контролером, а в режиме CLIENTSQL - нет. По этой причине, соединение типа CLIENTSQL можно применять для более интенсивных режимов регистрации и передачи данных. Рекомендованный интервал передачи данных на сервер в режиме CLIENTSQL составляет от 30 секунд и выше.

В режиме соединения типа CLIENTDB или CLIENTSQL контролер выступает в качестве клиента и поэтому можно применять обычные SIM-карты с поддержкой GPRS-соединения. Для успешного соединения с HTTP-сервером, контролер должен пройти аутентификацию, которая заключается в совпадении имени и пароля, которые задаются с помощью переменной ID. Кроме того, программа на языке ForthLogic™ должна обеспечить верный формат и фиксированное количество полей зарегистрированных данных для одной записи в файле регистрации, например:

"1304585524;22.233455;-1;13434", где:

1304585524	22.233455	-1	13434
Метка времени в формате UTC	Число с плавающей запятой, например температура	Флажок, например состояние входа	Целое число, например количество импульсов на входе

Разделителем между полями данных должен быть символ ";" (точка с запятой), в конце записи разделитель не применяется. Первым полем должно быть время в формате UTC. Первое поле является обязательным. Остальные поля должны строго отвечать параметрам соответствующей таблицы на сервере баз данных MYSQL.

В режиме соединения типа CLIENTDB или CLIENTSQL контролер также может получать обратную информацию в конце каждой сессии передачи файла на HTTP-сервер. Данная информация обрабатывается системой автоматически. При этом HTTP-сервер может свободно устанавливать и читать все глобальные переменные, глобальные переменные в формате чисел с плавающей запятой и глобальные битовые переменные. Программа на языке ForthLogic™ может воспользоваться данными в этих переменных и таким образом реализовать "обратную связь" с HTTP-сервером.

Примеры соединения типа SERVER и CLIENT

Приведем пример установки соединения типа SERVER. В этом примере используется SIM-карта с фиксированным IP-адресом и порт сервера по умолчанию (2020). Клиент должен устанавливать соединение с данным адресом и портом. Кроме того, клиент может быть только один и также должен иметь фиксированный IP-адрес.

```
 > ." static.umc.ua " SETAPN ." 192.168.0.1 " SETIP
(OK)
 > 0 CONSTANT checkservice_a
(OK)
 > : (checkservice) checkservice_a EXECUTE ;
```

```
(OK)
 > : reconfig SHUTTCP 3.0 1 TIMER! (checkservice) ;
(OK)
 > : task STATUSTCP NOT
 IF reconfig
 ELSE 1.0 1 TIMER! task
 THEN ;
(OK)
 > : openfail 3.0 1 TIMER! (checkservice) ;
(OK)
 > : start OPENTCP task openfail DROP ;
(OK)
 > : checkservice
 STATUSGPRS
 IF
   APN SERVER CONFIGTCP
   IF 3.0 1 TIMER! start
   ELSE 3.0 1 TIMER! reconfig
   THEN
 ELSE
   ATTACHGPRS DROP 10.0 1 TIMER! checkservice
 THEN ;
(OK)
 > ." checkservice " FIND TO checkservice_a
(OK)
 > : checknet
 SIGNAL? -1 =
 IF 1.0 1 TIMER! checknet
 ELSE 3.0 1 TIMER! checkservice
 THEN ;
(OK)
 > checknet
(OK)
 >
```

Сначала мы настроили параметры GPRS сервиса и позволили доступ только для клиента с IP-адресом 192.168.0.1. Слово "checknet" проверяет состояние GSM-модуля, когда он включится и успешно зарегистрируется в сети GSM, будет выполнено слово "checkservice", которое проверяет наличие GPRS сервиса и осуществляет попытку подключиться к сервису в случае его отсутствия. В случае его наличия происходит попытка подключить и настроить GPRS в режиме сервера. При успешной попытке будет выполнено слово "start". При неудачной попытке будет выполнено слово "reconfig", которое отключает GPRS и повторяет попытку путем выполнения слова "checkservice". Слово "start" активирует сервер и, в случае его успешной активации, запускает на выполнение слово "task", иначе все повторяется путем выполнения слова "checkservice". Слово "task" непрерывно проверяет состояние сервера и если он отключился (пропала сеть и тому подобное), то будет выполнено слово "reconfig", которое попробует возобновить работу сервера.

Теперь приведем пример установки соединения типа CLIENT.

```
 > ." static.umc.ua " SETAPN ." 192.168.0.100 2020 " SETIP
(OK)
 > 0 CONSTANT checkservice_a
(OK)
 > : (checkservice) checkservice_a EXECUTE ;
(OK)
 > : reconfig SHUTTCP 3.0 1 TIMER! (checkservice) ;
(OK)
 > : task STATUSTCP NOT
 IF reconfig
 ELSE 1.0 1 TIMER! task
 THEN ;
(OK)
 > : openfail 3.0 1 TIMER! (checkservice) ;
(OK)
 > : start IP OPENTCP task openfail DROP ;
(OK)
 > : checkservice
 STATUSGPRS
 IF
   APN CLIENT CONFIGTCP
   IF 3.0 1 TIMER! start
   ELSE 3.0 1 TIMER! reconfig
   THEN
 ELSE ATTACHGPRS DROP 10.0 1 TIMER! checkservice
 THEN ;
(OK)
 > ." checkservice " FIND TO checkservice_a
(OK)
 > : checknet
 SIGNAL? -1 =
 IF 1.0 1 TIMER! checknet
 ELSE 3.0 1 TIMER! checkservice
 THEN ;
(OK)
 > checknet
(OK)
 >
```

Сначала мы настроили параметры GPRS сервиса и указали параметры доступа к серверу. Остальные слова практически аналогичны предыдущему примеру. Лишь слово "start" осуществляет попытку подключиться к серверу, а слово "checkservice" настраивает режим клиента.

После установки соединения в любом обычном режиме CLIENT или SERVER, форт-система будет работать в GPRS режиме. При этом можно отправлять из программы строчные данные или данные регистрации (данные регистрации можно отсылать либо в процессе накопления, либо в виде файла сохраненного во внутренней памяти - см. раздел "Регистратор").

Модуль ETHERNET

Некоторые версии контролеров содержат встроенный порт ETHERNET и могут осуществлять обмен данными через локальную сеть ETHERNET или через INTERNET. На данный момент поддерживается работа по следующим протоколам: сервер MODBUS TCP в режиме подчиненного устройства (SLAVE), клиент передачи данных на сервер баз данных MYSQL с применением протокола HTTP.

Принципы клиент-серверного соединения ETHERNET

Необходимо детально остановиться на технологии клиент-сервер при передаче данных через локальную сеть ETHERNET или через INTERNET. На одном конце канала всегда находится *сервер* а на втором *клиент* и поэтому существует два сценария построения соединения. Первый сценарий заключается в том, что в качестве сервера удобно использовать серверную программу которая работает на ПК или на специализированном серверном оборудовании, которое имеет фиксированный IP-адрес во всемирной сети INTERNET. В этом случае контролер серии ES-ForthLogic™ со встроенным портом ETHERNET выступает в качестве клиента, который инициирует соединение с сервером. Преимущество такого сценария состоит в том, что к серверу одновременно может быть подсоединено очень много клиентов (сотни и тысячи) и для них нет необходимости в получении статических IP-адресов.

Второй сценарий заключается в том, что в качестве сервера выступает контролер серии ES-ForthLogic™ со встроенным портом ETHERNET, который может одновременно обработать лишь ограниченное количество клиентских соединений. При этом необходимо гарантировать доступность IP-адреса контролера во всемирной сети INTERNET - это достигается при применении статического IP-адреса. Обычно, такую услугу заказывают у интернет-провайдеров. Кроме этого, практически всегда возникает необходимость осуществить настройку локального сетевого оборудования для осуществления перенаправления портов. Преимущество данного сценария заключается в том, что на втором конце можно использовать клиентские программы, которые работают на ПК или другом оборудовании (планшет, смартфон и тому подобное) и имеют любой доступ к сети INTERNET.

Слова для работы с ETHERNET

Из программы на языке ForthLogic™ можно инициировать подключение к серверу в качестве клиента, разорвать соединение и посмотреть состояние соединения. Работа в качестве сервера происходит по большей части автоматически на уровне системы и абсолютно прозрачно для программы - можно лишь перезагрузить соответствующий сервер и посмотреть состояние соединения. Также из программы на языке ForthLogic™ можно осуществить общие настройки а также настройки работы соответствующего сервера. Для этого реализован ряд слов, которые представлены далее по тексту.

Начальные настройки

Начальные настройки контролеров, которые содержат встроенный порт ETHERNET, заключаются в установке параметров доступа к локальной сети ETHERNET аналогично тому, как это делается для других устройств в локальной сети ETHERNET (ПК, сетевого оборудования, принтеров и тому подобное). Слова, предназначенные для этого, описаны в разделе "Управление системой".

Для хранения параметров работы в режиме передачи данных на сервер баз данных MYSQL с применением протокола HTTP, в энергонезависимой памяти существуют специальные строчные переменные ID, IP и URL. Слова для работы с ними описаны в разделе "Слова для работы с GPRS".

Для выяснения существует ли физическое соединение с сетью ETHERNET, существует слово LAN? (состояние подключения к локальной сети) которое возвращает на стек логическое значение ИСТИНА, если существует физическое соединение с действующей сетью ETHERNET, или логическое значение НЕ ИСТИНА в противоположном случае.

Протокол сервера MODBUS TCP

Работа в качестве сервера MODBUS TCP происходит автоматически на уровне системы и абсолютно прозрачно для программы на языке ForthLogic™. Сервер MODBUS TCP работает в режиме подчиненного устройства MODBUS типа SLAVE и воспринимает соединение с клиентами, которые работают в режиме главного устройства MODBUS типа MASTER. Одновременно к серверу MODBUS TCP может быть присоединен лишь *один клиент*.

Принцип обмена данными с сервером MODBUS TCP заключается в том, что главное устройство MODBUS типа MASTER (в качестве которого может выступать как другой ПЛК так и система АСУТП или SCADA) делает запросы, а сервер MODBUS TCP (типа SLAVE) автоматически и прозрачно для пользователя осуществляет обработку запросов и формирование ответов. При этом, главное устройство MODBUS может свободно устанавливать и читать все глобальные переменные, глобальные переменные в формате чисел с плавающей запятой и глобальные битовые переменные.

С точки зрения главного устройства MODBUS типа MASTER, контролер представляется в виде устройства типа SLAVE с набором переменных расположенных в определенных адресах регистров. Регистр в интерпретации протокола MODBUS есть либо 1-битовая величина либо 16-битовая величина:

Адр	Название	Описание	Тип	Поручение
2000	VAR1	Глобальная переменная 1	длинный	03, 06, 16
2002	VAR2	Глобальная переменная 2	длинный	03, 06, 16
...	...	...	...	...
2508	VAR255	Глобальная переменная 255	длинный	03, 06, 16
2510	VAR256	Глобальная переменная 256	длинный	03, 06, 16
3000	FLOAT1	Глобальная переменная в формате чисел с плавающей запятой 1	плавающий	03, 06, 16
3002	FLOAT2	Глобальная переменная в формате чисел с плавающей запятой 2	плавающий	03, 06, 16
...	...	...	...	...
3252	FLOAT127	Глобальная переменная в формате чисел с плавающей запятой 127	плавающий	03, 06, 16
3254	FLOAT128	Глобальная переменная в формате чисел с плавающей запятой 128	плавающий	03, 06, 16
4000	FLAG1	Глобальная битовая переменная 1	бинарный	01, 05, 15
4001	FLAG2	Глобальная битовая переменная 2	бинарный	01, 05, 15
...	...	...	...	...

4254	FLAG255	Глобальная битовая переменная 255	бинарный	01, 05, 15
4255	FLAG256	Глобальная битовая переменная 256	бинарный	01, 05, 15

Все цифровые значения приведенные в таблице представлены в *десятичной форме*. *Адр* - это адрес регистра. *Поручение* - это код поручения (функции), которое может быть выполнено устройством типа SLAVE. Контролер может обслуживать коды следующих поручений:

- 01 (0x01) Read Coils (Чтение статуса дискретных выходов/регистров)
- 03 (0x03) Read Holding Registers (Чтение статуса регистров)
- 05 (0x05) Write Single Coil (Запись отдельного дискретного выхода/регистра)
- 06 (0x06) Write Single Register (Запись отдельного регистра)
- 15 (0x0F) Write Multiple Coils (Запись нескольких дискретных выходов/регистров)
- 16 (0x10) Write Multiple registers (Запись нескольких регистров)

Тип - это вид, в котором число записано в памяти контролера. *Бинарный тип* это число длинной в 1 бит, которое может принимать лишь два значения - "1" и "0", которые отвечают логическому состоянию ИСТИНА и НЕ ИСТИНА глобальных битовых переменных контролера. Переменная бинарного типа занимает 1 регистр MODBUS. *Длинный тип* это число длиной в четыре байта (32-бита) со знаком. Переменная длинного типа занимает 2 соседних регистра MODBUS и в таблице приведен адрес первого из них - в нем располагается старшее слово (16 бит) переменной. В переменной длинного типа возможно хранить целые числа в диапазоне от -2147483648 до +2147483647 и они полностью отвечают формату глобальных переменных контролера. *Плавающий тип* это число длиной в четыре байта (32-бита) согласно стандарту IEEE-754. Переменная плавающего типа занимает 2 соседних регистра MODBUS и в таблице приведен адрес первого из них - в нем располагается старшее слово (16 бит) переменной. В переменной плавающего типа возможно хранить числа с плавающей запятой одинарной точности согласно стандарту IEEE-754 и они полностью отвечают формату глобальных переменных с плавающей запятой контролера.

Программа на языке ForthLogic™ не имеет непосредственного доступа к обмену данными, однако может взаимодействовать с главным устройством MODBUS через глобальные переменные. Также можно перезагрузить сервер MODBUS TCP, посмотреть состояние соединения и настроить фильтрацию доступа клиентов на уровне IP-адресов.

В случае выявления сбоя в роботе сервера MODBUS TCP, его можно перезагрузить словом MBTCPRESET. При этом существующее соединение с клиентом будет разорвано и через пару секунд тот же клиент (или другой) может попробовать установить новое соединение.

Состояние соединения сервера MODBUS TCP с клиентом можно оценить с помощью слова MBTCPREMIP, которое печатает в выходном буфере и на терминале IP-адрес активного клиента. В частности, если он пустой - то в данный момент нет ни одного клиента подсоединенного к серверу MODBUS TCP.

Для обеспечения безопасности системы существует возможность настроить фильтрацию доступа клиентов к серверу MODBUS TCP на уровне IP-адресов. То есть, только клиент с *разрешенным* IP-адресом может установить соединение с сервером MODBUS TCP. Для установки разрешенного адреса предназначено слово MBTCPIPSET, которое работает следующим образом: из выходного буфера считывается строки - IP-адрес и запоминается в соответствующей ячейке внутренней энергонезависимой памяти. Для отображения разрешенного IP-адреса в выходном буфере и на терминале предназначено слово MBTCPIPGET. Если установить разрешенный адрес в виде пустой строки, то фильтрация отключатся вообще и любой клиент может установить соединение с сервером MODBUS TCP.

Протокол передачи данных на сервер баз данных MYSQL

При работе по протоколу передачи данных на сервер баз данных MYSQL, данные передаются автоматически из внутренней оперативной памяти на сервер баз данных MYSQL с применением протокола HTTP. Данный процесс начинается словом LOG>DBTCP и полностью контролируется системой.

Принцип передачи данных на сервер баз данных MYSQL заключается в том, что система сначала передает файл из внутренней оперативной памяти на HTTP-сервер с помощью метода POST. Параметры IP-адреса и номера порта HTTP-сервера необходимо указывать в переменной IP (подробно об этой переменной см. раздел "Слова для работы с GPRS"). Этот файл принимается и обрабатывается специализированным скриптом, имя которого необходимо указывать в переменной URL (подробно об этой переменной см. раздел "Слова для работы с GPRS"). Дальше этот файл анализируется и данные из него передаются в таблицу на сервер баз данных MYSQL. HTTP-сервер после получения файла не закрывает физическое соединение с контролером. По этой причине данный протокол можно применять для достаточно интенсивных режимов регистрации и передачи данных. Рекомендованный интервал передачи данных на сервер составляет от 5 секунд и выше.

Для успешного соединения с HTTP-сервером, контролер должен пройти аутентификацию, которая заключается в совпадении имени и пароля, которые задаются с помощью переменной ID (подробно об этой переменной см. раздел "Слова для работы с GPRS"). Кроме того, программа на языке ForthLogic™ должна обеспечить верный формат и фиксированное количество полей зарегистрированных данных для одной записи в файле регистрации, например:

"1304585524;22.233455;-1;13434", где:

1304585524	22.233455	-1	13434
Метка времени в формате UTC	Число с плавающей запятой, например температура	Флажок, например состояние входа	Целое число, например количество импульсов на входе

Разделителем между полями данных должен быть символ ";" (точка с запятой), в конце записи разделитель не применяется. Первым полем должно быть время в формате UTC. Первое поле является обязательным. Остальные поля должны строго отвечать параметрам соответствующей таблицы на сервере баз данных MYSQL.

При работе с данным протоколом контролер также может получать обратную информацию в конце каждой сессии передачи файла на HTTP-сервер. Данная информация обрабатывается системой автоматически. При этом HTTP-сервер может свободно устанавливать и читать все глобальные переменные, глобальные переменные в формате чисел с плавающей запятой и глобальные битовые переменные. Программа на языке ForthLogic™ может воспользоваться данными в этих переменных и таким образом реализовать "обратную связь" с HTTP-сервером.

Для установки соединения при работе по протоколу передачи данных на сервер баз данных MYSQL, существует слово DBTCPOPEN, которое работает следующим образом: из входного текста выбирается два очередных слова обратного вызова (введенных после слова DBTCPOPEN) и происходит настройка соединения. На стек при этом кладется логическое значение - результат выполнения слова DBTCPOPEN: логическое значение ИСТИНА означает, что процесс соединения начался успешно, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, уже существует соединение с сервером или отсутствует физическое соединение с сетью ETHERNET). Когда успешно установится соединение, форт-система, выполнит первое из указанных после слова DBTCPOPEN

слов обратного вызова, в случае, когда возникла ошибка при попытке соединения, форт-система, выполнит второе из указанных слов обратного вызова.

Для разрыва соединения при работе по протоколу передачи данных на сервер баз данных MYSQL в любой момент времени, существует слово DBTCPCLOSE. После разрыва соединения, можно осуществить попытку соединения с другим сервером. Для получения статуса соединения при работе по протоколу передачи данных на сервер баз данных MYSQL, существует слово DBTCPSTATUS, которое возвращает на стек логическое значение ИСТИНА если существует физическое соединение с удаленным сервером.

Приложение

Настройка программы-терминала

В качестве терминала в среде ОС Microsoft® Windows®XP, можно использовать программу эмуляции терминала Microsoft®HyperTerminal, которая входит в состав операционной системы.

Перед использованием программы эмуляции терминала, необходимо установить драйвер USB. Для этого необходимо либо запустить файл драйвера, который находится в папке "USB" дистрибутива, на выполнение, либо указать путь к папке "USB" дистрибутива (это зависит от версии контроллера). После успешной установки, в операционной системе появится новый последовательный порт (его номер можно проверить и поменять в программе "Диспетчер устройств").

Когда контролер подсоединен к компьютеру, можно запустить программу Microsoft®HyperTerminal (Пуск > Программы > Стандартные > Связь > HyperTerminal). При первом запуске, необходимо указать название нового подключения. Дальше в окне "Подключение" в списке "Подключаться через" выбрать номер последовательного порта, который был установлен драйвером USB. Дальше в окне "Свойства: COM" указать параметры связи: "Скорость" - 19200 для контролеров с дисплеем и 57600 для контролеров без дисплея; "Биты данных" - 8; "Четность" - нет; "Стоповые биты" - 1; "Управление потоком" - нет и утвердить все кнопкой "ОК". Дальше необходимо зайти в меню "Свойства" окна программы Microsoft®HyperTerminal (Файл > Свойства), выбрать вкладку "Параметры" и в разделе "Эмуляция терминала" выбрать "TTY", дальше нажать кнопку "Параметры ASCII", снять вся галочка и утвердить все настройки кнопкой "ОК". После этого, следует сохранить выбранные параметры, выполнив команду "Сохранить" в окне программы Microsoft®HyperTerminal (Файл > Сохранить).

Если запускать программу Microsoft®HyperTerminal через имя подключения, которое было выбрано при первом запуске, то все настройки автоматически вступают в силу и больше ничего настраивать не нужно.

Если программа Microsoft®HyperTerminal недоступна, то можно воспользоваться программой эмуляции терминала интегрированной среды разработки FLIDE - программой PUTTY. Параметры программы PUTTY уже настроены для работы с ПЛК серии ES-ForthLogic™, однако их можно произвольно менять (например шрифт, цвет, размер окна и тому подобное). Подробнее о программе PUTTY можно узнать из ее встроенной справки.

Создание звуковых файлов

Для реализации голосовых сообщений и меню, необходимо создать набор звуковых фрагментов и записать их в соответствующем формате на карту памяти SD/MMC. Для этого лучше всего подходит программа Microsoft®Звукозапись (Пуск > Программы > Стандартные > Развлечения > Звукозапись) из состава операционной системы Microsoft® Windows®XP. Каждый записанный звуковой фрагмент необходимо сохранить в файл с расширением "wav" в формате PCM с атрибутами "8,000 кГц; 8 бит; Моно 7КБ/с". Для этого в окне программы Microsoft®Звукозапись выбрать меню "Сохранить как..."

(Файл > Сохранить как...), в окне "Сохранение файла" в поле "Формат:" нажать кнопку "Изменить...", дальше в окне "Выбор звука" в поле "Формат:" выбрать PCM, а в поле "Атрибуты:" выбрать "8,000 кГц; 8 бит; Моно 7КБ/с" и утвердить все настройки кнопкой "ОК". Дальше указать название файла и сохранить его.

Сохраненые таким образом файлы, необходимо переписать в корневую директорию на карту памяти SD/MMC и вставить ее в контролер. В дальнейшем, при выполнении слова PLAY, данные файлы могут будут использованы для воспроизведения звуковых фрагментов.

Набор файлов чисел

Для воспроизведения словами чисел через аудиосистему контроллера, необходимо обеспечить на карте памяти SD/MMC набор *.wav файлов, которые должны иметь формат аналогичный описанному в предыдущем разделе и иметь фиксированные названия. В следующей таблице представлен перечень набора файлов.

Название файла	Голосовое содержимое файла	Название файла	Голосовое содержимое файла
_0.wav	нуль	_40.wav	сорок
_1.wav	один	_400.wav	четыреста
_10.wav	десять	_5.wav	пять
_100.wav	сто	_50.wav	пятьдесят
_11.wav	одиннадцать	_500.wav	пятьсот
_12.wav	двенадцать	_6.wav	шесть
_13.wav	тринадцать	_60.wav	шестьдесят
_14.wav	четырнадцать	_600.wav	шестьсот
_15.wav	пятнадцать	_7.wav	семь
_16.wav	шестнадцать	_70.wav	семьдесят
_17.wav	семнадцать	_700.wav	семьсот
_18.wav	восемнадцать	_8.wav	восемь
_19.wav	девятнадцать	_80.wav	восемьдесят
_2.wav	два	_800.wav	восемьсот
_20.wav	двадцать	_9.wav	девять
_200.wav	двести	_90.wav	девяносто
_3.wav	три	_900.wav	девятьсот
_30.wav	тридцать	comma.wav	запятая
_300.wav	триста	minus.wav	минус
_4.wav	четыре	plus.wav	плюс

Данный набор файлов должен находиться в папке с любым названием на карте памяти SD/MMC.

Создание файлов-скриптов на языке ForthLogic™

Непосредственно вводить с терминала большие тексты неудобно, поэтому их хранят в текстовых файлах и затем переносят в форт-систему. Эта возможность также может быть полезной при тиражировании скриптов-программ и для резервного сохранения созданного для конкретной задачи

скрипта-программы (это особенно актуально, когда осуществляется обновление программной прошивки контролера - при этом из словаря форт-системы удаляются все введенные пользователем слова).

Правила создания

Процедура создания файлов-скриптов чрезвычайно проста. В любом текстовом редакторе, например Microsoft®Блокнот (Пуск > Программы > Стандартные > Блокнот) из состава операционной системы Microsoft® Windows®XP, необходимо набрать текст скрипта и сохранить его в файл с расширением ".txt" или ".fl" (данное расширение является удобным для последующей работы с интегрированной средой разработки FLIDE и ассоциации файлов-скриптов с редактором среды). При этом необходимо следить, чтобы длина строки не превышала 77 символов и в конце файла была по крайней мере одна пустая строки. Кодировка символов в файле может быть любой кроме UNICODE.

Для большего удобства создания файлов-скриптов, можно воспользоваться специальным текстовым редактором интегрированной среды разработки FLIDE - программой Programmer's Notepad 2. В текстовом редакторе среды уже реализована подсветка синтаксиса языка ForthLogic™, визуальный контроль за максимальной длиной строки в файле и другие удобства для работы с текстовыми документами. Подробнее об этом можно узнать из встроенной в редактор справки.

Правила интерпретации с карты памяти

Если текст скрипта сохранить в корневой директории карты памяти SD/MMC в файле "forthdic.txt", то его можно интерпретировать в контролерах оборудованных дисплеем и считывателем карт памяти SD/MMC. Для этого нужно вставить карту в контролер и выполнить пункт конфигурационного меню "SD/MMC" > "ВЫПОЛНИТЬ:" > "ВЫП.СКРИПТ!". При этом данный файл будет построчно интерпретирован и выполнен текстовым интерпретатором форт-системы. При выявлении интерпретатором любой ошибки, его работа будет прекращена и на встроенном дисплее будет отображено сообщение о коде ошибки и номере строки файла, в которой возникла ошибка.

Если текст скрипта сохранить в корневой директории карты памяти SD/MMC в файле "autorun.txt", то при установке карты в любой контролер, оборудованный считывателем карт памяти SD/MMC, данный файл будет автоматически интерпретирован и выполнен текстовым интерпретатором форт-системы. При выявлении интерпретатором любой ошибки, его работа будет прекращена, потому важно, чтобы данный скрипт был предварительно отлажен в диалоговом режиме. После успешной интерпретации, данный файл будет автоматически *удален* из карты памяти.

Также для всех типов контролеров, оборудованных считывателем карт памяти SD/MMC, существует возможность интерпретации произвольного файла, расположенного на карте памяти SD/MMC. Для этого в терминальном режиме работы текстового интерпретатора форт-системы необходимо с новой строки указать команду COMPILE FILE <название файла>. При выявлении интерпретатором любой ошибки, его работа будет прекращена и в окне терминала будет отображено сообщение об ошибке и номере строки файла, в которой возникла ошибка.

Правила интерпретация через гипертерминал

Текстовый интерпретатор форт-системы может интерпретировать файлы не только из карты памяти SD/MMC. Если в терминальном режиме работы указать с новой строки команду RECEIVE FILE, то появится приглашение на пересылку текстового файла по протоколу CRC Xmodem. При этом, в течение 30 секунд необходимо инициировать отправку файла. В программе Microsoft®HyperTerminal для этого необходимо зайти в окно "Отправка файла" (Передача > Отправить файл...), выбрать с помощью кнопки "Обзор" имя файла, указать "Протокол"- Xmodem и нажать кнопку "Отправить". При выявлении интерпретатором любой ошибки, его работа будет прекращена, окно отправления файла

закроется и в окне терминала будет отображено сообщение об ошибке и номере строки файла, в которой возникла ошибка.

Интерпретация файлов через гипертерминал позволяет воспользоваться закодированными бинарными файлами скриптов (по умолчанию, для таких файлов принято расширение ".flx"). Такие файлы позволяют сохранить интеллектуальную собственность автора программы, ограничить количество инсталляции в контролеры и создаются специальным средствами. Процедура создания таких файлов предоставляется по требованию. Процедура интерпретации не отличается от обычной - необходимо лишь указывать команду RECEIVE FILEX.

Если программа Microsoft®HyperTerminal недоступна, то можно воспользоваться средствами пересылки файлов с интегрированной среды FLIDE - программой FLprog. FLprog это программа для передачи файлов-скриптов и закодированных бинарных файлов по протоколу CRC Xmodem в ПЛК серии ES-ForthLogic™. Программа не имеет собственного графического интерфейса и предназначенная для запуска с командной строки в составе разных интегрированных сред разработки программ (IDE), в частности в составе FLIDE.

Принципы эффективного программирования

При создании файлов-скриптов на языке ForthLogic™, следуют четко понимать отличие таких файлов от обычных файлов программ в других традиционных языках программирования. Файл-скрипт может состоять из определений новых слов и констант, а также из других слов и чисел, которые непосредственно обрабатываются и выполняются в *момент интерпретации* данного файла. Этот мощный механизм позволяет автоматизировать практически любую операцию (особенно с помощью карты памяти) и является главным отличием файлов-скриптов.

Создание программ

Написание программы, это по большей части интерактивный процесс: создание текста - проверка в реальных условиях - коррекция текста - проверка в реальных условиях - коррекция текста - ... Учитывая вышеупомянутую особенность файлов-скриптов, можно порекомендовать следующий алгоритм при написании и отлаживании программы:

1. Создать текстовый файл программы.
2. В начале файла добавить следующие строки:

```
STOPALL CLEARSYS
2 ERRORLEVEL
FORGET myprogram
0 ERRORLEVEL
: myprogram ." Короткое описание программы и ее версия " ;
```

3. Описать все необходимые слова и константы программы, при необходимости вставить слова, которые будут непосредственно выполнены во время интерпретации.
4. В конце файла описать "главное слово" программы (например *main*) и после описания добавить строки:

```
." main " BOOT
main
```

5. Добавить еще одну пустую строку и сохранить файл.
6. Отправить файл на интерпретацию через гипертерминал или через FLprog.
7. Если в процессе интерпретации не выявилось синтаксических или других ошибок, то программа будет запущена на выполнение и можно перейти к п.10.
8. При выявлении интерпретатором любой ошибки, его работа будет прекращена, окно отправления файла через гипертерминал закроется и в окне терминала будет отображено сообщение об ошибке и номере строки файла, в которой возникла ошибка. Если пересылка происходила через FLprog, то сообщение об ошибке будет отображено в окне FLprog.
9. После исправления ошибки необходимо повторить п.6.
10. После проверки работы программы, внести необходимые коррекции в файл и повторить п.6. Если вносить коррекции уже не нужно, то программа готова и файл является пригодным для передачи заказчику или для долговременного хранения.

И наконец, традиционная при написании программ рекомендация: максимально используйте комментарии в тексте файлов-скриптов (с помощью круглых скобок), разделяйте части программ пустыми строками и выделяйте отдельные конструкции программ отступами.

Отладка программ

Традиционные способы отладки программ заключаются в возможности остановки программы, просмотра и изменения контекста во время остановки, пошагового выполнения и тому подобное. В силу ограничений аппаратной платформы и особенности реализации многозадачности на языке ForthLogic™, не все классические средства и способы отладки возможны к реализации. Однако, интерпретирующий характер языка позволяет определенную степень свободы и предоставляет возможности недосягаемые для других программно-аппаратных платформ (главным образом основанных на компиляторах). Например, отсутствие этапа компиляция ускоряет процесс создания и отладки отдельных программных процедур (слов в терминах форт-системы): определив слово в терминальном режиме, его можно сразу проверить "подставив" через стек необходимые для его работы параметры. Доступ к интерпретатору в терминальном режиме во время работы программы также позволяет определенным образом вмешиваться в ее работу: стеки и глобальные переменные доступны для модификации в произвольный момент времени. Однако, для дополнительного удобства, существуют слова которые позволяют более эффективно осуществлять процесс отладки. Рассмотрим их более подробно.

Для динамического отображения состояния выполнения программы используется специальный строчный *буфер сообщений* длиной до 78 символов, содержание которого отображается красным цветом в квадратных скобках во время работы в режиме отображения статусной информации о системе в окне терминала (см. раздел "Работа в диалоговом режиме"). В любом месте программы можно применить слово MESSAGE (сообщение), которое записывает текст из выходного буфера в буфер сообщений и он становится видимым в режиме отображения статусной информации. Таким образом можно организовать передачу данных из программы о тех или других процессах, которые происходят во время исполнения. Текст в буфере сообщений можно дописывать. Для этого его нужно копировать в выходной буфер с помощью слова MESSAGE. (сообщение-точка), добавить любым способом новый текст к содержимому выходного буфера и опять записать результат в буфер сообщений.

Чтобы "увидеть" содержимое обоих стеков не изменяя их содержимого, используются слова .S (точка стек) и .FS (точка математический стек), которые соответственно печатают на терминале содержимое целочисленного и математического стеков в квадратных скобках в том порядке, как это принято для

стековой нотации - верхние значения на стеке (те которые были добавлены последними) находятся правее. Значения с математического стека печатаются в научном представлении с 6 знаками после десятичной запятой. Данные слова можно использовать лишь в режиме интерпретации (их нельзя использовать при определении других слов).

Реализация многозадачности с помощью таймеров является простой и мощной концепцией. Иногда бывает необходимо остановить все задачи, которые привязаны к разным таймерам. Вручную останавливать каждый таймер долго и не эффективно. Это можно сделать с помощью слова STOPALL (остановить все). При этом обнуляются промежутки времени и слова обратного вызова всех таймеров, останавливаются все обмены по протоколу MODBUS RTU, останавливается любая автоматическая деятельность на линиях входов/выходов, останавливается регистрация в режиме пользователя, останавливается воспроизведение сообщений и любая активность в канале GSM/GPRS/CSD, останавливается работа клиентов в канале ETHERNET. Стеки, глобальные переменные и статическое состояние выходных сигналов остаются без изменений. Для обнуления программно доступных стеков, глобальных переменных, буфера сообщений и выходных сигналов, существует слово CLEARSYS (очистить систему). Более "глубокую очистку" можно выполнить с помощью перезапуска форт-системы. Слово RESTART (перезапуск) выполняет перезагрузку всей форт-системы и аналогично холодному старту (выключение/включение питания контроллера). Данные слова можно использовать как в режиме интерпретации так и в составе других слов.

Для реализации механизма точек остановки программы, существуют слова BREAKPOINT (точка остановки) и DEBUGLEVEL (уровень отладки). Слово BREAKPOINT предназначено для остановки форт-системы в процессе отладки и работает следующим образом: со стека снимается верхнее значение - номер точки остановки и, если этот номер меньше или равен глобальной системной переменной DEBUGLEVEL, то происходит остановка форт-системы. При этом останавливаются все таймеры, останавливаются все обмены по протоколу MODBUS RTU, останавливается любая автоматическая деятельность на линиях входов/выходов, останавливается регистрация в режиме пользователя, останавливается воспроизведение сообщений и любая активность в канале GSM/GPRS/CSD, останавливается работа клиентов в канале ETHERNET, а на терминале в фигурных скобках печатается информация о номере точки остановки, содержимое всех стеков (целочисленного, математического и возвратов) и выходного буфера на момент остановки. После этого все стеки и буфера обнуляются а глобальные переменные остаются без изменений.

Слово DEBUGLEVEL предназначено для установки одноименной системной переменной и работает следующим образом: со стека снимается верхнее значение и запоминается в глобальной системной переменной DEBUGLEVEL. Начальное значение этой переменной равняется нулю и она не является энергонезависимой. Содержание данной переменной можно интерпретировать как *уровень отладки*, который активирует разные группы точек остановки. Например, если в программе существует несколько групп точек остановки, которые имеют номера от 100 до 200 и от 200 до 300, то присвоение системной переменной DEBUGLEVEL значения 200 блокирует вторую группу точек остановки (форт-система их игнорирует), а присвоение системной переменной DEBUGLEVEL значения 0 (значение по умолчанию) блокирует все точки остановки и программа работает в обычном режиме. Таким образом, слова BREAKPOINT могут оставаться в штатной программе постоянно при условии, что их номера больше нуля. Слово DEBUGLEVEL можно использовать лишь в режиме интерпретации.

Кроме точек остановки можно применять точки наблюдения. Для этого в любом фрагменте текста программы необходимо применить слово WATCHPOINT (точка наблюдения). При достижении этого слова со стека снимается верхнее значение - номер точки наблюдения и форт-система печатает на терминале в квадратных скобках текущее время и номер точки, а в круглых скобках содержимое целочисленного и математического стеков а также выходного буфера на данный момент. После этого форт-система продолжает выполнение программы.

В процессе создания программы возможны определенные типы ошибок которые приводят к перезапуску всей системы. Если в системе настроен авто запуск главного слова (с помощью меню или

слова BOOT), то при возникновении вышеупомянутых ошибок система будет непрерывно перезагружаться. Для избежания такого явления реализован механизм блокирования выполнения главного слова в терминальном режиме работы. То есть, при подключенном кабеле USB и работе через терминал авто запуск не происходит. В этом случае запустить программу можно вручную набрав ее главное слово в терминале.

Ошибки языка ForthLogic™

В терминальном и дистанционном режиме работы форт-системы, все ошибки, которые возникают во время диалога с системой, по умолчанию отображаются в круглых скобках в виде текста на английском языке. Во время интерпретации файла "forthdic.txt", в случае возникновения ошибки, на дисплее будет отображено сообщение о коде ошибки, а не сам текст. Дополнительно по умолчанию, в терминальном режиме работы в квадратных скобках отображаются все ошибки, которые возникают во время работы всей форт-системы (это касается и тех частей, которые выполняют слова назначенные таймерам и другим отложенным во времени процессам так и части, которая интерпретирует слова в процессе диалога). При этом, в квадратных скобках отображается время возникновения ошибки, текст ошибки и расшифрованное состояние стека возвратов на момент возникновения ошибки в виде цепочки слов - данная информация позволяет реконструировать место возникновения ошибки. Отображение ошибок по умолчанию можно отключать и включать с помощью слова ERRORLEVEL (уровень отображения ошибок), которое снимает со стека числовой параметр *уровня отображения ошибок* и настраивает соответственное поведение форт-системы. Данный параметр не сохраняется во время пропадания питания и по умолчанию равняется 0 - что означает отображение всех ошибок. Если установить данный параметр равным 1, то отображаются только ошибки в квадратных скобках в терминальном режиме работы. Если установить данный параметр равным 2, то форт-система вообще не отображает никаких ошибок - такой режим может пригодиться в некоторых случаях во время отладки программы.

Далее представлены все известные форт-системе ошибки, их коды и объяснения:

Код	Текст ошибки	Объяснение
1	UNKNOWN WORD	Неизвестное форт-системе слово
2	ILLEGAL USAGE	Применение слова в неверном контексте (выполнение или компиляция)
3	ILLEGAL PARAMETER	Неверный параметр слова (вне разрешенных пределов)
4	INSUFFICIENT PARAMETERS	Недостаточное количество параметров
5	DATA STACK EMPTY	Стек данных пустой
6	DATA STACK FULL	Переполнение стека данных
7	RETURN STACK EMPTY	Стек возвратов пустой
8	RETURN STACK FULL	Переполнение стека возвратов
9	OUT OF MEMORY	Недостаточно памяти в словаре
10	MATHEMATIC STACK EMPTY	Математический стек пустой
11	MATHEMATIC STACK FULL	Переполнение математического стека
12	SD CARD NOT FOUND	Карта памяти SD/MMC отсутствует
13	FILE NOT FOUND	Файл не найден
14	INPUT BUFFER OVERFLOW	Переполнение входного буфера
15	FILE SYSTEM BUSY	Файловая система занята (например, воспроизводится звуковой файл)

Код	Текст ошибки	Объяснение
16	FILE EMPTY	Файл пустой
17	WRONG CONSTRUCTION	Неполная конструкция управления выполнением (IF-ELSE-THEN)
18	FILE TRANSFER ERROR	Ошибка во время приема файла по протоколу Xmodem
19	FILE TRANSFER TIMOUT	Инициированный прием файла по протоколу Xmodem не начался в течение 30 сек.
20	COMPILE MODE NOT ALLOWED	Запрещение режима компиляции (касается текстов SMS)

Аппаратная платформа языка ForthLogic™

Аппаратная платформа, на которой реализована конкретная форт-система, с точки зрения программной модели характеризуется разным количеством программно доступных ресурсов и разным набором базовых слов.

Для всех аппаратных платформ общими являются следующие ресурсы и параметры. Все целочисленные арифметические и логические операции осуществляются над стеком данных глубиной 16 элементов. Диапазон представления целых чисел от -2147483648 до 2147483647. Все математические операции с числами с плавающей запятой осуществляются над математическим стеком глубиной 16 элементов. Диапазон представления чисел с плавающей запятой отвечает одинарной точности согласно стандарту IEEE-754: $\pm(1.4x10^{-45}...3.4x10^{38})$. В логических операциях ИСТИНА означает произвольное число не ровное 0 (обычно -1), НЕ ИСТИНА означает 0. Операции, которые возвращают логическое значение ИСТИНА, возвращают его в виде -1. Информация в выходном буфере дублируется или отображением в окне терминала в терминальном режиме форт-системы или отображением в тексте SMS в дистанционном режиме форт-системы для аппаратных платформ, которые имеют модуль GSM. Глубина программно недосягаемого стека возвратов равняется 64 уровням. Стек возвратов используется системой для рекурсивного вызова встроенных слов из других встроенных слов а также для хранения контекста конструкции IF-ELSE-THEN.

В следующей таблице версий представлены ресурсы форт-системы, которые зависят от аппаратной платформы и версии прошивок, для которых также актуальна данная инструкция по программированию:

Ресурс	Короткое описание	Стр.	B	D	G	полная	SG	SE	SB	ST	M
FIRMWARE	Версия прошивки (микропрограммы)		1.16(B)	1.16(D)	1.16(GII)	4.50	2.06(SG) 1.14(SG)	3.06(SB)	2.06(SB)	2.06(ST)	1.02(M)
D_MAX	Количество глобальных переменных для хранения целых чисел		64	64	64	128	256	256	256	128	64
F_MAX	Количество глобальных математических переменных для хранения чисел в формате с плавающей запятой		32	32	32	64	128	128	128	64	16
B_MAX	Количество глобальных битовых переменных		128	128	128	128	256	256	256	128	64
S_MAX	Количество строчных переменных		-	4	4	8	16	4	4	8	-
T_MAX	Количество независимых таймеров		32	32	32	64	128	128	128	64	32
DI_MAX	Количество цифровых входов		8	8	8	8	6*	6*	6*	6*	6*
AI_MAX	Количество аналоговых входов		4	4	4	4	2	2	2	2	-
DO_MAX	Количество цифровых выходов		4	4	4	4	6*	6*	6*	6*	6*
RO_MAX	Количество релейных выходов		3	3	3	3	-	-	-	-	-
MENU_MAX	Количество пунктов меню пользователя		-	4	-	8	-	-	-	-	-
ROW_MAX	Количество строк текста дисплея		-	7	-	6	-	-	-	-	-
COL_MAX	Количество столбцов текста дисплея		-	15	-	15	-	-	-	-	-
PHONE_MAX	Количество переменных для хранения номеров телефонов		-	-	40	40	50	-	-	50	-
OUTBUF_MAX	Размер выходного буфера, байт		120	120	180	180	180	120	120	180	80
LOGBUF_MAX	Размер буфера регистрации, байт		-	-	512	256	1024	2048	2048	-	-

* из состава универсальных цифровых входов/выходов. Например если задействовано 4 входа то остается 2 выхода, если задействован 1 вход, то остается 5 выходов, и так далее.

Встроенные слова языка ForthLogic™

Далее представлены все известные форт-системе встроенные слова с коротким описанием и ссылкой на соответствующую страницу документа с детальным описанием:

№	Условная группа	Слово	Короткое описание	Стр.	B	D	G	полная	SG	SE	SB	ST	M
1	вх/вых	AI?	кладет на мат. стек состояние аналогового входа		+	+	+	+	+	+	+	+	
2	вх/вых	AIS?	кладет на мат. стек масштабируемое состояние аналогового входа					+					
3	вх/вых	BAT?	кладет на мат. стек напряжение питания аккумулятора		+	+	+	+					
4	вх/вых	POW?	кладет на мат. стек напряжение основного питания		+	+	+	+					
5	вх/вых	DI?	кладет на стек состояние цифрового входа		+	+	+	+	+	+	+	+	+
6	вх/вых	DO!	устанавливает состояние цифрового выхода равным значению со стека		+	+	+	+	+	+	+	+	+
7	вх/вых	DO?	кладет на стек состояние цифрового выхода		+	+	+	+	+	+	+	+	+
8	вх/вых	RO!	устанавливает состояние релейного выхода равным значению со стека		+	+	+	+					
9	вх/вых	RO?	кладет на стек состояние релейного выхода		+	+	+	+					
10	вх/вых	WIEGANDSTART	настраивается и запускается ядро протокола WIEGAND-26						+	+	+		
11	вх/вых	WIEGANDSTOP	останавливается ядро протокола WIEGAND-26						+	+	+		
12	вх/вых	PWMSTART	настраивается и запускается генерация сигнала ШИМ						+	+	+		
13	вх/вых	PWMSTOP	останавливается генерация сигнала ШИМ						+	+	+		
14	вх/вых	COUNTERSTART	настраивается и запускается счетчик импульсов						+	+	+		
15	вх/вых	COUNTERSTOP	останавливается счетчик импульсов						+	+	+		
16	вх/вых	COUNTERGET	на стек возвращается содержимое счетчика импульсов						+	+	+		
17	GSM	OPERATOR	печать в вых. буфере оператора сети				+	+	+			+	
18	GSM	ROAMING?	кладет на стек признак работы в роуминге				+	+	+			+	
19	GSM	SIGNAL?	кладет на стек уровень сигнала сети				+	+	+			+	
20	GSM	MODULE?	кладет на стек расширенные параметры сети и модуля GSM						+			+	
21	GSM	SIM?	кладет на стек состояние активации карты SIM					+	+			+	
22	GSM	LAST	печатает в вых. буфере номер телефона последнего абонента				+	+	+			+	
23	GSM	USER	печатает в вых. буфере номер телефона пользователя				+	+	+			+	
24	GSM	USERPHONE	устанавливает номер телефона пользователя				+	+	+			+	
25	GSM/текст	SMS	отправляет SMS				+	+	+			+	
26	GSM/текст	USSD	отправление USSD-запрос				+	+	+			+	
27	GSM/голос	ANSWER	ответ на голосовой звонок					+	+				
28	GSM/голос	CLIP	выполнение действия в ответ на голосовой звонок				+	+	+			+	
29	GSM/голос	DIAL	осуществление голосового звонка					+	+				
30	GSM/голос	HOLD	прекращение голосового звонка					+	+				
31	GSM/голос	HOOK?	кладет на стек статус голосового звонка					+	+				
32	GSM/голос	MUTE	прекращение воспроизведения звука голосового меню					+	+				
33	GSM/голос	PLAY	воспроизведение звукового файла с карты памяти SD/MMC					+	+				
34	GSM/голос	SAY	воспроизведение значения с вершины мат. стека					+	+				
35	GSM/голос	MIC	подключение к каналу GSM внешней аудио системы и динамика					+	+				
36	GSM/голос	VOICE	подключение к каналу GSM внутреннего синтезатора					+	+				
37	GSM/голос	MICLEVEL	настройка чувствительности внешней аудио системы					+	+				
38	GSM/голос	SPKLEVEL	настройка усиления внешнего динамика					+	+				
39	GSM/голос	AUTOSAYPLUS	включает автоматическое воспроизведение знака плюс					+	+				
40	GSM/голос	NOAUTOSAYPLUS	выключает автоматическое воспроизведение знака плюс					+	+				
41	GSM/CSD	CONNECTCSD	создание канала данных CSD				+		+				
42	GSM/CSD	SENDCSD	отправление строчных данных через канал CSD				+		+				
43	GSM/CSD	BREAKCSD	разрыв канала данных CSD				+		+				
44	GSM/CSD	STATUSCSD	кладет на стек состояние канала данных CSD				+		+				
45	GSM/CSD	MODEMCSD	активирует прозрачный модемный режим				+		+				
46	GSM/DTMF	TONE	генерация сигналов DTMF					+	+				
47	GSM/DTMF	WAITKEY	введение одиночных сигналов DTMF					+	+				
48	GSM/DTMF	WAITPW	введение пароля с помощью сигналов DTMF					+	+				
49	GSM/DTMF	WAITSTR	введение строки цифр с помощью сигналов DTMF					+	+				
50	GSM/DTMF	DTMFCONFIRM	включает автоматическое подтверждение принятого сигнала DTMF					+	+				
51	GSM/DTMF	NODTMFCONFIRM	выключает автоматическое подтверждение принятого сигнала DTMF					+	+				
52	GSM/GPRS	APN	печатает в вых. буфере содержимое энергонезависимой переменной APN				+		+				
53	GSM/GPRS	SETAPN	устанавливает содержимое энергонезависимой переменной APN				+		+				
54	GSM/GPRS	IP	печатает в вых. буфере содержимое энергонезависимой переменной IP				+		+				
55	GSM/GPRS	SETIP	устанавливает содержимое энергонезависимой переменной IP				+		+				
56	GSM/GPRS	ID	печатает в вых. буфере содержимое энергонезависимой переменной ID				+		+				
57	GSM/GPRS	SETID	устанавливает содержимое энергонезависимой переменной ID				+		+				
58	GSM/GPRS	URL	печатает в вых. буфере содержимое энергонезависимой переменной URL						+				
59	GSM/GPRS	SETURL	устанавливает содержимое энергонезависимой переменной URL						+				
60	GSM/GPRS	ATTACHGPRS	подключение к сервису GPRS				+		+				
61	GSM/GPRS	CONFIGTCP	настройка GPRS соединения				+		+				
62	GSM/GPRS	CLIENT	системная константа для слова CONFIGTCP				+		+				
63	GSM/GPRS	CLIENTDB	системная константа для слова CONFIGTCP						+				
64	GSM/GPRS	CLIENTSQL	системная константа для слова CONFIGTCP						+				
65	GSM/GPRS	CLIENTOPC	системная константа для слова CONFIGTCP				+		+				
66	GSM/GPRS	SERVER	системная константа для слова CONFIGTCP				+		+				
67	GSM/GPRS	CONFIGPORT	настройка порта в режиме сервера				+		+				
68	GSM/GPRS	DNSMODE	настройка режима URL-адрес				+		+				
69	GSM/GPRS	IPMODE	настройка режима IP-адрес				+		+				
70	GSM/GPRS	OPENTCP	настройка соединения согласно протокола TCP				+		+				
71	GSM/GPRS	CLOSETCP	разрыв соединение согласно протокола TCP				+		+				
72	GSM/GPRS	SHUTTCP	отключение от GPRS				+		+				
73	GSM/GPRS	SENDTCP	передача строчных данных согласно протокола TCP				+		+				
74	GSM/GPRS	STATUSGPRS	возвращает на стек состояние сервиса GPRS				+		+				
75	GSM/GPRS	STATUSSERVER	возвращает на стек состояние работы в режиме сервера				+		+				
76	GSM/GPRS	STATUSTCP	возвращает на стек состояние соединения согласно протокола TCP				+		+				
77	GSM/GPRS	STATUSOPC	возвращает на стек состояние соединения с OPC-сервером				+		+				

№	Условная группа	Слово	Короткое описание	Стр.	B	D	G	полная	SG	SE	SB	ST	M
78	GSM/GPRS	REMOTEIP	печатает в вых. буфере IP-адрес клиента в режиме сервера				+		+				
79	GSM/GPRS	LOCALIP	печатает в вых. буфере собственный IP-адрес				+		+				
80	GSM/GPRS	M2MTCPON	включает режим без ответа форт-системы				+		+				
81	GSM/GPRS	M2MTCPOFF	включает нормальный режим ответа форт-системы				+		+				
82	ETHERNET	LAN?	кладет на стек состояние соединения с сетью ETHERNET							+			
83	ETHERNET	MBTCPRESET	перезагрузка сервера MODBUS TCP							+			
84	ETHERNET	MBTCPREMIP	печатает в вых. буфере адрес активного клиента MODBUS TCP							+			
85	ETHERNET	MBTCPIPSET	устанавливает адрес разрешенного клиента MODBUS TCP							+			
86	ETHERNET	MBTCPIPGET	печатает в вых. буфере адрес разрешенного клиента MODBUS TCP							+			
87	ETHERNET	DBTCPOPEN	настройка соединения с сервером передачи данных							+			
88	ETHERNET	DBTCPCLOSE	разрыв соединение с сервером передачи данных							+			
89	ETHERNET	DBTCPSTATUS	возвращает на стек состояние соединения с сервером передачи данных							+			
90	меню	FOCUS	устанавливает фокус на пункт меню пользователя			+		+					
91	меню	HIDE	делает невидимым пункт меню пользователя			+		+					
92	меню	INFO	печатает текст в информационном поле меню пользователя					+					
93	меню	LASTMENU?	кладет на стек номер последнего задействованного пункта меню пользователя			+		+					
94	меню	MENU	настраивает и делает видимым пункт меню пользователя			+		+					
95	дисплей	BLACK	изменение цвета шрифта на черный			+		+					
96	дисплей	BLUE	изменение цвета шрифта на голубой			+		+					
97	дисплей	DEEPBLUE	изменение цвета шрифта на синей			+		+					
98	дисплей	GREEN	изменение цвета шрифта на зеленый			+		+					
99	дисплей	RED	изменение цвета шрифта на красный			+		+					
100	дисплей	ORANGE	изменение цвета шрифта на оранжевый			+		+					
101	дисплей	VIOLET	изменение цвета шрифта на фиолетовый			+		+					
102	дисплей	WHITE	изменение цвета шрифта на белый			+		+					
103	дисплей	YELLOW	изменение цвета шрифта на желтый			+		+					
104	дисплей	INVERT	инверсия цветов шрифта и фона			+		+					
105	дисплей	CLEAR	очистка дисплея			+		+					
106	дисплей	PRINT	печать строки с выходного буфера на дисплее			+		+					
107	дисплей	GET	отображение на дисплее окна для введения числовых значений			+		+					
108	дисплей	GETS	отображение на дисплее окна для введения строчных значений			+		+					
109	клавиши	BUTTON	настраивает функцию клавиши			+		+					
110	клавиши	F1	системная константа для слов BUTTON, LED! и LED?			+		+		+	+		+
111	клавиши	F2	системная константа для слов BUTTON, LED! и LED?			+		+		+	+		
112	клавиши	DOWN	системная константа для слова BUTTON			+		+					
113	клавиши	LEFT	системная константа для слова BUTTON			+		+					
114	клавиши	OK	системная константа для слова BUTTON			+		+					
115	клавиши	RIGHT	системная константа для слова BUTTON			+		+					
116	клавиши	UP	системная константа для слова BUTTON			+		+					
117	сигнал	BEEP	генерация звукового сигнала		+	+	+	+	+	+	+	+	
118	сигнал	LED!	устанавливает состояние встроенного светодиода равным значению со стека							+	+		+
119	сигнал	LED?	кладет на стек состояние встроенного светодиода							+	+		+
120	аудио	AUDIOMUTE	прекращение воспроизведения звука через аудиосистему						+				
121	аудио	AUDIOPLAY	воспроизведение звук. файла с карты SD/MMC через аудиосистему						+				
122	аудио	AUDIOSAY	воспроизведение значения с вершины мат. стека через аудиосистему						+				
123	сист. время	>DATE	превращает время UTC в общепринятую дату			+	+	+	+	+	+	+	
124	сист. время	>TIME	превращает время UTC в общепринятое время			+	+	+	+	+	+	+	
125	сист. время	>UTC	превращает общепринятое время во время UTC		+	+	+	+	+	+	+	+	
126	сист. время	>WDAY	превращает время UTC в общепринятый день недели			+	+	+	+	+	+	+	
127	сист. время	DATE?	кладет на стек значение годов, месяцев, дней локального времени		+	+	+	+	+	+	+	+	
128	сист. время	TIME?	кладет на стек значение секунд, минут, часов локального времени		+	+	+	+	+	+	+	+	
129	сист. время	UTC?	кладет на стек секунды UTC так как это принято в ОС UNIX		+	+	+	+	+	+	+	+	
130	сист. время	WDAY?	кладет на стек значение дней недели		+	+	+	+	+	+	+	+	
131	сист. время	DST?	кладет на стек признак действия летнего времени				+		+	+	+	+	
132	сист. время	TIMESTAMP	печатает в вых. буфере календарное представление времени				+		+	+	+	+	
133	сист. время	SUNRISE?	кладет на стек значение секунд, минут, часов восхода солнца						+	+	+		
134	сист. время	SUNSET?	кладет на стек значение секунд, минут, часов заката солнца						+	+	+		
135	сист. время	CIVIL	системная константа для слов SUNRISE? и SUNSET?						+	+	+		
136	сист. время	OFFICIAL	системная константа для слов SUNRISE? и SUNSET?						+	+	+		
137	MODBUS	MODBUSSTART	формируется и активируется пакет протокола MODBUS RTU		+	+	+	+	+	+	+		
138	MODBUS	MODBUSSTOP	прекращается формирование пакета протокола MODBUS RTU		+	+	+	+	+	+	+		
139	MODBUS	CYCLIC_ACCESS	системная константа для слова MODBUSSTART		+	+	+	+	+	+	+		
140	MODBUS	SINGLE_ACCESS	системная константа для слова MODBUSSTART		+	+	+	+	+	+	+		
141	MODBUS	READ_COILS	системная константа для слова MODBUSSTART		+	+	+	+	+	+	+		
142	MODBUS	READ_HOLDREGS	системная константа для слова MODBUSSTART		+	+	+	+	+	+	+		
143	MODBUS	READ_INPUTREGS	системная константа для слова MODBUSSTART		+	+	+	+	+	+	+		
144	MODBUS	READ_INPUTS	системная константа для слова MODBUSSTART		+	+	+	+	+	+	+		
145	MODBUS	WRITE_COIL	системная константа для слова MODBUSSTART		+	+	+	+	+	+	+		
146	MODBUS	WRITE_COILS	системная константа для слова MODBUSSTART		+	+	+	+	+	+	+		
147	MODBUS	WRITE_REG	системная константа для слова MODBUSSTART		+	+	+	+	+	+	+		
148	MODBUS	WRITE_REGS	системная константа для слова MODBUSSTART		+	+	+	+	+	+	+		
149	MODBUS	MODBUSCALLBACK	настраивает автоматический обратной вызов протокола MODBUS RTU		+	+	+	+	+	+	+		
150	MODBUS	MODBUSPARAM	устанавливает параметры последовательного канала протокола MODBUS RTU		+	+	+	+	+	+	+		
151	MODBUS	EVEN	системная константа для слова MODBUSPARAM		+	+	+	+	+	+	+		
152	MODBUS	NONE	системная константа для слова MODBUSPARAM		+	+	+	+	+	+	+		
153	MODBUS	ODD	системная константа для слова MODBUSPARAM		+	+	+	+	+	+	+		
154	MODBUS	MODBUSSTATUS?	состояние обмена по протоколу MODBUS RTU		+	+	+	+	+	+	+		
155	MODBUS	MODBUSTIMEOUT!	устанавливает время TIMEOUT протокола MODBUS RTU		+	+	+	+	+	+	+		
156	MODBUS	MODBUSRESET	перезапуск ядра протокола MODBUS RTU						+	+	+		
157	регистр.	LOGON	запускает и настраивает процесс регистрации		+	+	+	+	+	+	+		
158	регистр.	LOGRUN	запускает процесс регистрации с существующими настройками		+	+	+	+	+	+	+		
159	регистр.	LOGOFF	останавливает процесс регистрации		+	+	+	+	+	+	+		
160	регистр.	ALL_DATA	системная константа для слова LOGON		+	+	+	+	+	+	+		
161	регистр.	EVENTS_MODE	системная константа для слова LOGON		+	+	+	+	+	+	+		

№	Условная группа	Слово	Короткое описание	Стр.	B	D	G	полная	SG	SE	SB	ST	M
162	регистр.	USER_MODE	системная константа для слова LOGON		+	+	+	+	+	+	+		
163	регистр.	INPUTS	системная константа для слова LOGON		+	+	+	+	+	+	+		
164	регистр.	INTERVAL_MODE	системная константа для слова LOGON		+	+	+	+	+	+	+		
165	регистр.	OUTPUTS	системная константа для слова LOGON		+	+	+	+	+	+	+		
166	регистр.	TO_FLASH	системная константа для слова LOGON		+	+	+	+	+	+	+		
167	реестр	TO_RAM	системная константа для слова LOGON						+	+	+		
168	регистр.	TO_SD	системная константа для слова LOGON		+	+	+	+	+	+	+		
169	регистр.	TO_TCP	системная константа для слова LOGON				+		+				
170	регистр.	LOG	переписывает выходной буфер в системный реестр		+	+	+	+	+	+	+		
171	регистр.	LOG?	кладет на стек состояние процесса регистрации		+	+	+	+	+	+	+		
172	регистр.	LOGSEND?	кладет на стек состояние процесса переноса данных		+	+	+	+	+	+	+		
173	регистр.	LOGFILE?	кладет на стек статус зарегистрированных данных		+	+	+	+	+	+	+		
174	регистр.	LOG>SD	переносит данные регистрации на карту памяти SD/MMC		+	+	+	+	+	+	+		
175	регистр.	LOG>TCP	переносит данные регистрации через канал GPRS				+		+				
176	регистр.	LOG>DB	переносит данные регистрации через канал GPRS в базу MYSQL						+				
177	регистр.	LOG>SQL	переносит данные регистрации через канал GPRS в базу MYSQL (потоковый реж.)						+				
178	регистр.	LOG>DBTCP	переносит данные регистрации через канал ETHERNET в базу MYSQL							+			
179	регистр.	FREELOG?	кладет на стек количество свободного места во внутр. памяти регистрации		+	+	+	+	+	+	+		
180	регистр.	LOGFORMAT	форматирует внутреннюю память регистрации		+	+	+	+	+	+	+		
181	регистр.	LOGFILETOGLE	переключение активного файла реестра						+	+	+		
182	регистр.	LOGFILEDELETE	удаление не активного файла реестра						+	+	+		
183	регистр.	LOGTITLEOFF	выключает автоматическое оглавление во время регистрации		+	+	+	+	+	+	+		
184	регистр.	LOGTITLEON	включает автоматическое оглавление во время регистрации		+	+	+	+	+	+	+		
185	регистр.	LOGTERMINON	выключает автоматическое окончание строки во время регистрации				+		+	+	+		
186	регистр.	LOGTERMINOFF	включает автоматическое окончание строки во время регистрации				+		+	+	+		
187	служебн.	:	начало определения нового слова		+	+	+	+	+	+	+	+	+
188	служебн.	;	окончание определения нового слова		+	+	+	+	+	+	+	+	+
189	служебн.	CONSTANT	определяет числовую константу		+	+	+	+	+	+	+	+	+
190	служебн.	TO	записывает вершину стека в числовую константу		+	+	+	+	+	+	+	+	+
191	служебн.	FCONSTANT	определяет числовую математическую константу		+	+	+	+	+	+	+	+	+
192	служебн.	TOF	записывает вершину мат. стека в числовую математическую константу		+	+	+	+	+	+	+	+	+
193	служебн.	ARRAY	определяет массив числовых констант						+	+	+	+	
194	служебн.	SET	записывает вершину стека в элемент массива						+	+	+	+	
195	служебн.	FINDINDEX	поиску элемента массива равного определенному значению						+	+	+	+	
196	служебн.	IF	слово условного оператора управления выполнением алгоритма		+	+	+	+	+	+	+	+	+
197	служебн.	ELSE	слово условного оператора управления выполнением алгоритма		+	+	+	+	+	+	+	+	+
198	служебн.	THEN	слово условного оператора управления выполнением алгоритма		+	+	+	+	+	+	+	+	+
199	служебн.	EXECUTE	выполнение исполнительного адреса с вершины стека		+	+	+	+	+	+	+	+	+
200	служебн.	FIND	кладет на стек исполнительный адрес слова из выходного буфера		+	+	+	+	+	+	+	+	+
201	служебн.	NAME	отображение в вых. буфере слова заданного исполнительным адресом		+	+	+	+	+	+	+	+	+
202	служебн.	WORDS	отображение на экране в терминальном режиме словаря		+	+	+	+	+	+	+	+	+
203	служебн.	FORGET	удаление слова из словаря		+	+	+	+	+	+	+	+	+
204	служебн.	ERASE	удаление всех слов из словаря					+	+	+	+	+	
205	служебн.	UNUSED	кладет на стек свободное место в словаре в байтах		+	+	+	+	+	+	+	+	+
206	служебн.	RESTART	перезапуск системы		+	+	+	+	+	+	+	+	+
207	служебн.	CLEARSYS	очистка системы		+	+	+	+	+	+	+	+	+
208	служебн.	STOP	слово, которое ничего не выполняет (но оооочень полезное!)		+	+	+	+	+	+	+	+	+
209	служебн.	STOPALL	останавливает все таймеры		+	+	+	+	+	+	+	+	+
210	служебн.	TIMER!	запуск таймера, который после заданного времени выполнит заданное слово		+	+	+	+	+	+	+	+	+
211	служебн.	TIMER?	кладет на стек адрес слова, которое будет выполнено таймером		+	+	+	+	+	+	+	+	+
212	служебн.	AUTOSPACE	режим отображения в вых. буфере с автоматическим пробелом		+	+	+	+	+	+	+	+	
213	служебн.	NOAUTOSPACE	режим отображения в вых. буфере без автоматического пробела		+	+	+	+	+	+	+	+	
214	служебн.	FPREC!	настройка точности отображения мат. стека		+	+	+	+	+	+	+	+	
215	служебн.	VERSION	отображение в вых. буфере версии прошивки		+	+	+	+	+	+	+	+	+
216	службов	SYSINFO	отображение в вых. буфере расширенной версии прошивки						+	+	+	+	
217	служебн.	(	начало комментария, конец или символ) или конец строки		+	+	+	+	+	+	+		+
218	служебн.	.FS	неразрушающее отображение на терминале всего мат. стека		+	+	+	+	+	+	+	+	+
219	служебн.	.S	неразрушающее отображение на терминале всего стека		+	+	+	+	+	+	+	+	+
220	служебн.	BREAKPOINT	точка остановки форт-системы		+	+	+	+	+	+	+	+	
221	служебн.	DEBUGLEVEL	устанавливает уровень отладки для точки остановки		+	+	+	+	+	+	+	+	
222	служебн.	ERRORLEVEL	устанавливает уровень отображения ошибки		+	+	+	+	+	+	+	+	+
223	служебн.	WATCHPOINT	точка наблюдения форт-системы						+	+	+	+	
224	служебн.	MESSAGE	устанавливает содержание буфера сообщений						+	+	+	+	+
225	служебн.	MESSAGE.	печатает в вых. буфере содержание буфера сообщений						+	+	+	+	+
226	конфигур.	BOOT	устанавливает скрипт авто запуска системы		+	+	+	+	+	+	+	+	+
227	конфигур.	BOOT.	печатает в вых. буфере скрипт авто запуска системы		+	+	+	+	+	+	+	+	+
228	конфигур.	CALBAT	калибровка аккумуляторной батареи		+		+						
229	конфигур.	CALI	калибровка комбинированного входа по току		+		+		+	+	+	+	
230	конфигур.	CALPOW	калибровка напряжения питания		+		+						
231	конфигур.	CALV	калибровка комбинированного входа по напряжению		+		+						
232	конфигур.	CALZ	калибровка нуля комбинированного входа						+	+	+	+	
233	конфигур.	PASSWORD	устанавливает системный пароль		+		+		+	+	+	+	+
234	конфигур.	PROTECTPARAM	возвращает на стек параметры защиты системы				+		+	+	+	+	+
235	конфигур.	PASSWORD.	печатает в вых. буфере системный пароль				+		+	+	+	+	+
236	конфигур.	DISABLE	системная константа для слова PASSWORD		+		+		+	+	+	+	+
237	конфигур.	PROTECT_BY	системная константа для слова PASSWORD		+		+		+	+	+	+	+
238	конфигур.	CONTROL	устанавливает права доступа для дистанционного режима				+		+			+	
239	конфигур.	CONTROLPARAM	возвращает на стек параметры доступа для дистанционного режима				+		+			+	
240	конфигур.	FOR_ALL	системная константа для слова CONTROL				+		+			+	
241	конфигур.	FOR_LOYAL	системная константа для слова CONTROL				+		+			+	
242	конфигур.	LOCAL	системная константа для слова CONTROL				+		+			+	
243	конфигур.	REMOTE	системная константа для слова CONTROL				+		+			+	
244	конфигур.	PIN	устанавливает PIN-код				+		+			+	
245	конфигур.	PIN.	печатает в выходной буфер PIN-код				+		+			+	

№	Условная группа	Слово	Короткое описание	Стр.	B	D	G	полная	SG	SE	SB	ST	M
246	конфигур.	DIAI	устанавливает параметры комбинированного входа		+		+						
247	конфигур.	SET_TO_D	системная константа для слова DIAI		+		+						
248	конфигур.	SET_TO_I	системная константа для слова DIAI		+		+						
249	конфигур.	SET_TO_V	системная константа для слова DIAI		+		+						
250	конфигур.	DIAIPARAM	возвращает на стек параметры комбинированного входа		+		+						
251	конфигур.	SETWATCH	устанавливает параметры системных часов		+		+		+	+	+	+	
252	конфигур.	SUMMER_OFF	системная константа для слова SETWATCH		+		+		+	+	+	+	
253	конфигур.	SUMMER_ON	системная константа для слова SETWATCH		+		+		+	+	+	+	
254	конфигур.	WATCH!	устанавливают системные часы		+	+	+	+	+	+	+	+	
255	конфигур.	WATCHPARAM	возвращает на стек параметры системных часов				+		+	+	+	+	
256	конфигур.	ETHIP	устанавливает собственный IP-адрес							+			
257	конфигур.	ETHIP.	печатает в выходной буфер собственный IP-адрес							+			
258	конфигур.	ETHGATEWAY	устанавливает IP-адрес основного шлюза							+			
259	конфигур.	ETHGATEWAY.	печатает в выходной буфер IP-адрес основного шлюза							+			
260	конфигур.	ETHMASK	устанавливает маску подсети							+			
261	конфигур.	ETHMASK.	печатает в выходной буфер маску подсети							+			
262	конфигур.	MBTCPPORT!	устанавливает с вершины стека порт сервера MODBUS TCP							+			
263	конфигур.	MBTCPPORT?	кладет на стек порт сервера MODBUS TCP							+			
264	строки	."	отображение в вых. буфере строки		+	+	+	+	+	+	+	+	+
265	строки	FLUSH	очистка выходного буфера		+	+	+	+	+	+	+	+	
266	строки	LENGTH	кладет на стек длину строки текста в вых. буфере			+	+	+	+	+	+	+	
267	строки	NEWLINE	перенесение текста в вых. буфере на новую строку			+	+	+	+	+	+	+	
268	строки	SPACE	отображение в вых. буфере пробела		+	+	+	+	+	+	+	+	
269	строки	QUOTE	отображение в вых. буфере кавычки		+	+	+	+	+	+	+	+	
270	строки	STRING!	записывает в строчную переменную текст из выходного буфера			+	+	+	+	+	+	+	
271	строки	STRING?	кладет в выходной буфер строку текста из строчной переменной			+	+	+	+	+	+	+	
272	строки	STR>VAR	переносит символы из выходного буфера в целочисленную переменную						+	+	+		
273	мат.опер.	F-	вычитание на математическом стеке		+	+	+	+	+	+	+	+	+
274	мат.опер.	F*	умножение на математическом стеке		+	+	+	+	+	+	+	+	+
275	мат.опер.	F**	функция возведения в степень на математическом стеке		+	+	+	+	+	+	+	+	
276	мат.опер.	F.	отображение в вых. буфере вершины мат. стека в формате с фикс. запятой		+	+	+	+	+	+	+	+	+
277	мат.опер.	F/	деление на математическом стеке		+	+	+	+	+	+	+	+	+
278	мат.опер.	F+	сложение на математическом стеке		+	+	+	+	+	+	+	+	+
279	мат.опер.	F<	логическая операция МЕНЬШЕ на математическом стеке		+	+	+	+	+	+	+	+	+
280	мат.опер.	F=	логическая операция РАВНО на математическом стеке						+	+	+	+	
281	мат.опер.	F>	логическая операция БОЛЬШЕ на математическом стеке		+	+	+	+	+	+	+	+	+
282	мат.опер.	F<=	логическая операция МЕНЬШЕ РАВНО на математическом стеке						+	+	+	+	
283	мат.опер.	F<>	логическая операция НЕ РАВНО на математическом стеке						+	+	+	+	
284	мат.опер.	F>=	логическая операция БОЛЬШЕ РАВНО на математическом стеке						+	+	+	+	
285	мат.опер.	FABS	получение абсолютного значения на математическом стеке		+	+	+	+	+	+	+	+	
286	мат.опер.	FACOS	функция арккосинус на математическом стеке		+	+	+	+	+	+	+	+	
287	мат.опер.	FASIN	функция арксинус на математическом стеке		+	+	+	+	+	+	+	+	
288	мат.опер.	FATAN	функция арктангенс на математическом стеке		+	+	+	+	+	+	+	+	
289	мат.опер.	FCOS	функция косинус угла на математическом стеке		+	+	+	+	+	+	+	+	
290	мат.опер.	FCOSH	функция косинус гиперболический на математическом стеке		+	+	+	+	+	+	+	+	
291	мат.опер.	FDEPTH	кладет на стек количество элементов мат. стека		+	+	+	+	+	+	+	+	+
292	мат.опер.	FDROP	отбрасывание вершины мат. стека		+	+	+	+	+	+	+	+	+
293	мат.опер.	FDUP	дублирование вершины мат. стека		+	+	+	+	+	+	+	+	+
294	мат.опер.	FE.	отображение в вых. буфере вершины мат. стека в научном формате		+	+	+	+	+	+	+	+	+
295	мат.опер.	FEXP	функция экспонента на математическом стеке		+	+	+	+	+	+	+	+	
296	мат.опер.	FLN	функция логарифм натуральный на математическом стеке		+	+	+	+	+	+	+	+	
297	мат.опер.	FLOG	функция логарифм десятичный на математическом стеке		+	+	+	+	+	+	+	+	
298	мат.опер.	FNEGATE	изменение знака числа на математическом стеке		+	+	+	+	+	+	+	+	
299	мат.опер.	FOVER	дублирование значение мат. стеке под вершиной на вершину мат. стека		+	+	+	+	+	+	+	+	+
300	мат.опер.	FPICK	дублирование произвольного значения мат. стека на вершину мат. стека		+	+	+	+	+	+	+	+	+
301	мат.опер.	FROLL	циклическая ротация произвольного количества верхних значений матем. стека		+	+	+	+	+	+	+	+	+
302	мат.опер.	FROT	циклическая ротация трех верхних значений мат. стека		+	+	+	+	+	+	+	+	+
303	мат.опер.	FSIN	функция синус угла на математическом стеке		+	+	+	+	+	+	+	+	
304	мат.опер.	FSINH	функция синус гиперболический на математическом стеке		+	+	+	+	+	+	+	+	
305	мат.опер.	FSQRT	функция корень квадратный на математическом стеке		+	+	+	+	+	+	+	+	
306	мат.опер.	FSWAP	перестановка двух верхних значений мат. стека		+	+	+	+	+	+	+	+	+
307	мат.опер.	FTAN	функция тангенс угла на математическом стеке		+	+	+	+	+	+	+	+	
308	мат.опер.	FTANH	функция тангенс гиперболический на математическом стеке		+	+	+	+	+	+	+	+	
309	мат.опер.	FVAR!	записывает в математическую переменную значение с мат. стека		+	+	+	+	+	+	+	+	+
310	мат.опер.	FVAR?	кладет на мат. стек значение математической переменной		+	+	+	+	+	+	+	+	+
311	цел.опер.	-	операция вычитания		+	+	+	+	+	+	+	+	+
312	цел.опер.	*	операция умножения		+	+	+	+	+	+	+	+	+
313	цел.опер.	.	отображение в вых. буфере вершины стека		+	+	+	+	+	+	+	+	+
314	цел.опер.	/	операция деления с отбрасыванием остатка		+	+	+	+	+	+	+	+	+
315	цел.опер.	+	операция добавления		+	+	+	+	+	+	+	+	+
316	цел.опер.	<	логическая операция МЕНЬШЕ		+	+	+	+	+	+	+	+	+
317	цел.опер.	<=	логическая операция МЕНЬШЕ РАВНО		+	+	+	+	+	+	+	+	+
318	цел.опер.	<>	логическая операция НЕ РАВНО		+	+	+	+	+	+	+	+	+
319	цел.опер.	=	логическая операция РАВНО		+	+	+	+	+	+	+	+	+
320	цел.опер.	>	логическая операция БОЛЬШЕ		+	+	+	+	+	+	+	+	+
321	цел.опер.	>=	логическая операция БОЛЬШЕ РАВНО		+	+	+	+	+	+	+	+	+
322	цел.опер.	ABS	операция получения абсолютного значения		+	+	+	+	+	+	+	+	+
323	цел.опер.	AND	операция логического И		+	+	+	+	+	+	+	+	+
324	цел.опер.	DEPTH	кладет на стек количество элементов стека данных		+	+	+	+	+	+	+	+	+
325	цел.опер.	DROP	отбрасывание вершины стека		+	+	+	+	+	+	+	+	+
326	цел.опер.	DUP	дублирование вершины стека		+	+	+	+	+	+	+	+	+
327	цел.опер.	FALSE	кладет на стек логическое значение НЕ ИСТИНА		+	+	+	+	+	+	+	+	+
328	цел.опер.	FLAG!	записывает в битовую переменную значение со стека		+	+	+	+	+	+	+	+	+
329	цел.опер.	FLAG?	кладет на стек значение битовой переменной		+	+	+	+	+	+	+	+	+

№	Условная группа	Слово	Короткое описание	Стр.	B	D	G	полная	SG	SE	SB	ST	M
330	цел.опер.	ISNOW	попарно сравнивает шесть значений на вершине стека		+	+	+	+	+	+	+	+	
331	цел.опер.	LSHIFT	операция логического сдвига влево		+	+	+	+	+	+	+	+	+
332	цел.опер.	MOD	операция деления с получением остатка		+	+	+	+	+	+	+	+	+
333	цел.опер.	NEGATE	операция изменения знака числа		+	+	+	+	+	+	+	+	+
334	цел.опер.	NOT	операция логического ОТРИЦАНИЯ		+	+	+	+	+	+	+	+	+
335	цел.опер.	OR	операция логического ИЛИ		+	+	+	+	+	+	+	+	+
336	цел.опер.	OVER	дублирование значение под вершиной стека на вершину стека		+	+	+	+	+	+	+	+	+
337	цел.опер.	PICK	дублирование произвольного значения стека на вершину стека		+	+	+	+	+	+	+	+	+
338	цел.опер.	ROLL	циклическая ротация произвольного количества верхних значений на стеке		+	+	+	+	+	+	+	+	+
339	цел.опер.	ROT	циклическая ротация трех верхних значений на стеке		+	+	+	+	+	+	+	+	+
340	цел.опер.	RSHIFT	операция логического сдвига вправо		+	+	+	+	+	+	+	+	+
341	цел.опер.	SWAP	перестановка двух верхних значений стека		+	+	+	+	+	+	+	+	+
342	цел.опер.	TRUE	кладет на стек логическое значение ИСТИНА		+	+	+	+	+	+	+	+	+
343	цел.опер.	VAR!	записывает в переменную значение со стека		+	+	+	+	+	+	+	+	+
344	цел.опер.	VAR?	кладет на стек значение переменной		+	+	+	+	+	+	+	+	+
345	цел.опер.	XOR	операция логического ИСКЛЮЧАЮЩЕГО ИЛИ		+	+	+	+	+	+	+	+	+
346	пр.типов	D>F	перенесение вершины стека данных на вершину мат. стека		+	+	+	+	+	+	+	+	+
347	пр.типов	F>D	перенесение вершины мат. стека на вершину стека данных		+	+	+	+	+	+	+	+	+
348	пр.типов	F>US	превращение "математический формат в без знаковый"		+	+	+	+	+	+	+		
349	пр.типов	US>F	превращение "без знаковый формат в математический"		+	+	+	+	+	+	+		
350	пр.типов	US>DF	превращение "без знаковый формат в двойной математический"						+	+	+		
351	пр.типов	DF>US	превращение "двойной математический формат в без знаковый"						+	+	+		
352	пр.типов	US>S	превращение "без знаковый формат в знаковый"		+	+	+	+	+	+	+		

Внесенные изменения и дополнения

Версия документа 2.3

- добавлен раздел "Передача данных GPRS";
- добавлен раздел "Воспроизведение сообщений через акустическую систему";
- в приложении добавлен раздел "Набор файлов чисел";
- в разделе "Последовательный порт RS485" дополнено описание слова MODBUSPARAM;
- в раздел "Управление системой" добавлено описание слов WATCHPARAM, CALZ, CONTROLPARAM, PROTECTPARAM, PASSWORD. (пароль-точка), PIN. (пин-точка);
- в раздел "Системное время" добавлено описание слов DST?, TIMESTAMP;
- в разделе "Воспроизведение сообщений" дополнено описание слова SAY;
- в раздел " Воспроизведение сообщений" добавлено описание слов AUTOSAYPLUS, NOAUTOSAYPLUS;
- в раздел "Передача данных CSD" добавлено описание слова MODEMCSD;
- в раздел "Регистратор" добавлено описание слов TO_TCP и LOG>TCP;
- в таблице ресурсов и таблице слов добавлен столбик версии контролера ES-ForthLogic-SG.

Версия документа 2.4

- в разделе "Последовательный порт RS485" дополнено описание слова MODBUSTIMEOUT!;
- в разделе "Передача данных GPRS" добавлено описание слов M2MTCPON, M2MTCPOFF, ATTACHGPRS и дополнено описание слов SETIP, SETAPN;
- в разделе "Регистратор" добавлено описание слов USER_MODE, LOGSEND?, LOGFILE? и дополнено описание слов LOG?, LOG;
- в разделе "Создание программ" изменено описание образца файла-скрипта;
- в разделе "Отладка программ" добавлено описание слова CLEARSYS;
- в разделе "Ошибки языка ForthLogic™" добавлено описание слова ERRORLEVEL;

Версия документа 2.5

- в разделе "Система" добавлено описание команды RESTORE DEFAULTS;
- в разделе "Константы, строки и переменные" добавлено описание слова QUOTE;
- в разделе "Регистратор" дополнено описание слова LOGFILE?;
- в разделе "Управления системой" дополнено описание слов SETWATCH и PASSWORD;
- в разделе "Управления системой" добавлено описание слова DIAIPARAM;

Версия документа 2.6

- в разделе "Регистратор" добавлено описание слова LOG>DB и дополнено описание процесса регистрации;
- раздел "Акустическая система" дополнено и переименовано на "Гарнитура";
- в разделе "Передача данных GPRS" добавлено описание слов ID, URL, SETID, SETURL, DNSMODE, IPMODE, CLIENTDB, CLIENTOPC, STATUSOPC

Версия документа 2.7

- добавлен новый раздел "Индикация"
- добавлен новый раздел "Идентификация системы"
- добавлен новый раздел "Модуль ETHERNET"

- дополнен раздел "Создания файлов-скриптов на языке ForthLogic™"
- в разделе "Константы, строка и переменная" добавлено описание слов ARRAY, SET, FINDINDEX
- в разделе "Отладка программ" добавлено описание слов MESSAGE, MESSAGE, WATHCPOINT и изменен синтаксис слова BREAK на BREAKPOINT.
- в разделе "Системное время" добавлено описание слов SUNRISE?, SUNSET?, CIVIL, OFFICIAL
- в разделе "Последовательный порт RS485" добавлен описание слов MODBUSRESET, US>DF, DF>US, STR>VAR
- в разделе "Слова для работы с GPRS" добавлено описание слов CLIENTSQL
- в разделе "Регистратор" добавлено описание слов TO_RAM, LOG>SQL, LOG>DBTCP, LOGTERMINON, LOGTERMINOFF, LOGFILETOGLE, LOGFILEDELETE
- в разделе "Введение новых слов" добавлено описание слов ERASE
- в разделе "Логические операции" добавлено описание слов F=, F<=, F<>, F>=
- в разделе "Состояние сети GSM" добавлено описание слов MODULE?, SIM?
- в разделе "Входы" добавлено описание слов WIEGANDSTART, WIEGANDSTOP, COUNTERSTART, COUNTERSTOP, COUNTERGET
- в разделе "Выходы" добавлено описание слов PWMSTART, PWMSTOP
- в разделе "Управление системой" добавлено описание слов ETHIP, ETHGATEWAY, ETHMASK,ETHIP., ETHGATEWAY., ETHMASK., MBTCPPORT!, MBTCPPORT?
- в приложении "Аппаратная платформа языка ForthLogic" добавлено описание новых контролеров;
- в приложении "Встроенные слова языка ForthLogic" добавлено описание новых контролеров;